FPGA IMPLEMENTATION OF AUTO TV TRACKING ALGORITHMS

By

Manvendra Singh

DEPARTMENT OF ELECTRICAL ENGINEERING

Indian Institute of Technology Kanpur

FEBRUARY, 2002

FPGA IMPLEMENTATION OF AUTO TV TRACKING ALGORITHMS

A Thesis Submitted in
Partial Fulfillment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

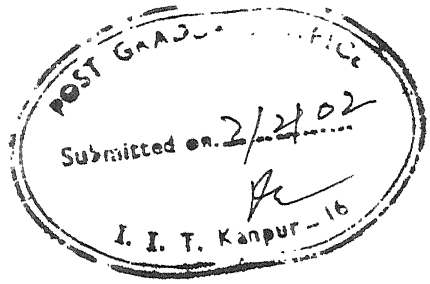
MANVENDRA SINGH

to the

DEPARTMENT OF
ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY
KANPUR

February 2002

CERTIFICATE



This is to certify that the work containing in the thesis entitled **"FPGA IMPLEMENTATION OF AUTO TV TRACKING ALGORITHMS"** by **MANVENDRA SINGH** has been carried out under my supervision and that this work has not been submitted elsewhere for the award of a degree.

Dr. BAQUER MAZHARI
Assistant Professor,
Department of Electrical Engineering,
Indian Institute of Technology, Kanpur.

February 2002.

ACKNOWLEDGMENT

I express my sincere and deep gratitude to my thesis supervisor Dr. B. Mazhari for his inspiring guidance and encouragement at every stage of this work. He provided me with numerous valuable, innovative and constructive suggestions, without which this work would not have been possible. No words can aptly describe his patience and understanding. I would also like to thank all those instructors who have taught me various courses during the academic program with special mention of Dr. R. Sharan who has been throughout a source of encouragement and guidance.

I am thankful to Instrument Research & Development Establishment (I.R.D.E.) of D.R.D.O. for providing me this wonderful opportunity. The support from my colleagues of this establishment has been very valuable. The students who have done this course with me have also been extremely helpful. I am grateful to all of them.

I am extremely grateful to my parents, relatives and friends for their love, support, blessings and encouragement. My wife Manju deserves a special mention here for her patience and support.

February 2002

Manvendra Singh

ABSTRACT

Real-time target tracking in video sequences is an essential component of any Electro-optical system. The auto TV tracker computes the target position in successive frames with respect to the center of the field-of-view (FOV) using image tracking techniques. In the present work centriod and correlation based tracking algorithms have been discussed, designed and implemented into FPGAs using hardware description languages (i.e. VHDL). The centroid algorithm has been designed using loop pipelining method for real-time application. This design approach has been successfully simulated and implemented into Xilinx's FPGA XC4008E. For the implementation of correlation based tracking algorithm, three-memory approach has been used to cater the real-time requirement. The correlation based tracking uses the Sequential Similarity Detection Algorithm (SSDA) for the calculation of match point as it require less hardware than the classical algorithm. This technique has been designed, simulated and successfully implemented into Xilinx's FPGA XC4020E. The advantage of using FPGA implementation is that it can work as a stand-alone system and may be integrated with other system by using the standard communication protocol. This design approach has given the system, much-needed reconfiguribilty to suit the customer's requirement. It has also resulted in reduction of PCB fabrication time since the hardware remains essentially the same although the system design may change.

CONTENTS

List of Figures ---------- vii

Chapter 1. An Introduction to Auto TV tracker ---------- 1

1.1 Introduction ---------- 1

1.1.1 Need of TV Tracker ---------- 1

1.1.2 Working Principle ---------- 2

1.1.3 TV Tracker Specification ---------- 4

1.1.4 Tracking Algorithms ---------- 5

1.1.5 Performance Characteristics of TV Tracker -------- 7

1.1.6 Pre-processing ---------- 8

1.2 Objective of Thesis ---------- 11

1.3 Organization of Work ---------- 11

Chapter 2. Centroid Based Tracking Algorithm ---------- 13

2.1 Background ---------- 13

2.2 FPGA Based System Design ---------- 15

2.3 Design of Centroid Based Tracking Algorithm ---------- 16

2.3.1 Generation of Manual Track Gate Size ---------- 18

2.3.2 Generation of Manual Track Gate Size ---------- 19

2.3.3 Computational Logic ---------- 20

2.4 Results and Discussion ---------- 23

Chapter 3. Correlation Based Tracking Algorithm ---------- 26

3.1 Background ---------- 26

3.2 System Configuration of Correlation Based Tracking ---- 29

3.3 FPGA implementation of Correlation Based Tracking ---- 30

3.4 Results and Discussion ---------- 34

Conclusion ---------- 39

Future Scope ---------- 40

Bibliography ---------- 41

Appendix – I ---------- 43

Appendix – II ---------- 64

List of Figures

1.1 System Configuration of Auto TV tracker ----------3

1.2 Schematic Diagram of Sync-separator circuit ----------9

1.3 Waveform of threshold white and black hot target data ----------10

2.1 Scheme of partitioning the active area ----------13

2.2 TV picture (binary target representation) ----------14

2.3 Flow diagram for FPGA based design methodology ----------16

2.4 Scheme of centroid tracking implementation ----------17

2.5 Scheme for manual track gate size ----------18

2.6 Scheme for auto track gate size ----------19

2.7 Scheme for implementation of computational logic----------20

2.8 Waveform of control signals used in the computational logic ----------22

2.9 Scheme of divider logic ----------23

2.10 Simulation result of the divider implemented ----------24

2.11 Simulation result of the computational logic implemented ----------25

3.1 Geometry of correlating reference image with search image ----------27

3.2 System configuration of correlation based tracking ----------30

3.3 FPGA implementation of correlation tracking algorithm ----------32

3.4 Interconnect logic for correlation tracker ----------33

3.5 Scheme for the implementation of memory controller ----------33

3.6 Scheme of computational logic for correlation ----------34

3.7 Simulation result of address generator ----------35

3.8 Simulation result of address generator showing the completion of 16 x 16 block ----------36

3.9 Simulation result of interconnect logic ----------36

3.10 Simulation result of ADC input data ----------37

3.11 Simulation result of computational logic (SSDA method) ----------37

CHAPTER 1

An Introduction to Auto TV Tracker

1.1 Introduction:

A typical TV tracker system consists a TV camera, monitor and servo mechanism. The operator analyzes the image coming from camera. If the operator sees some target of interest on monitor then he/she will follow the target through the manual control switch (i.e. joystick).

This type of TV systems suffered from a significant number of problems such as:

- The necessity of having an operator to monitor the video signal, compromises overall system capabilities.
- The TV system is "dead weight" when an operator is not monitoring the video.

To ease the operator work and incorporate some intelligence in the system, an automatic TV tracker is used. The Auto TV tracker process the contrast information contained in the video signal obtained from TV camera (either CCD or Thermal Imager) and generates the horizontal and vertical error signals using the tracking algorithms. These error signals are further used to drive servo system on which the sensors are mounted.

1.1.1 Need of TV Tracker

The auto TV tracker has wide range of applications in defense systems. In defense system it is used in almost all the systems, where the video monitor is provided to the operator for the analysis of scene. It is used in land defense system such as Main Battle Tank (MBT) for the alignment of Gunner's main- sight to give the accurate hitting of target. The TV tracker is also used in

the missile-seeker heads, Remotely Piloted Vehicles (RPV's) as well as in conventional aircraft and helicopters. It is widely used in the test ranges to track the missile path trajectory during the missile test fire. If suitable range data is provided through Laser Range Finder (LRF) then auto TV tracker alongwith the LRF could be used in automatic Fire Control System (FCS).

The auto TV tracker is a key component in the Electro-Optical Fire Control Fire Control System (EOFCS) for naval ships. Where it provides the continuous azimuth (Az) and elevation (El) position error data of the target of interest based upon the video image supplied by the EOFCS sensors. Using these positional data provided by the TV tracker, FCS compute the fire trajectory that finally goes to the Gun Control System for accurate hitting.

1.1.2 Working principle:

The Typical auto TV tracker system configuration is shown in the fig.(1.1). An auto TV tracker receives a video input, locks onto a target selected by the operator within the TV field-of-view (FOV) and outputs error signals, describing the displacement of the target from the center of the field-of-view. A TV formatted system [either CCD, Thermal Imager (FLIR) etc.] produces a video signal in a sequential manner. The exact scanning format depends upon the sensor design (for this case it is of CCIR-B slandered i.e. 625 lines with 20 ms frame rate). In most sensors scanning begins at the upper left, sweeps to the right, continuing line-by-line down through the scene till the end of frame. By detecting the beginning of each line (i.e. horizontal sync), the end of the frame (i.e. vertical sync), and by using a high speed clock, the video signal may be broken into an X-Y coordinate matrix (i.e. Az, El) of TV lines. Now by choosing the suitable tracking algorithm such as (Edge, Centroid or Correlation, the auto TV tracker computes the azimuth and elevation position signal derived from the video contrast information of the selected target within the sensor field-of-view.

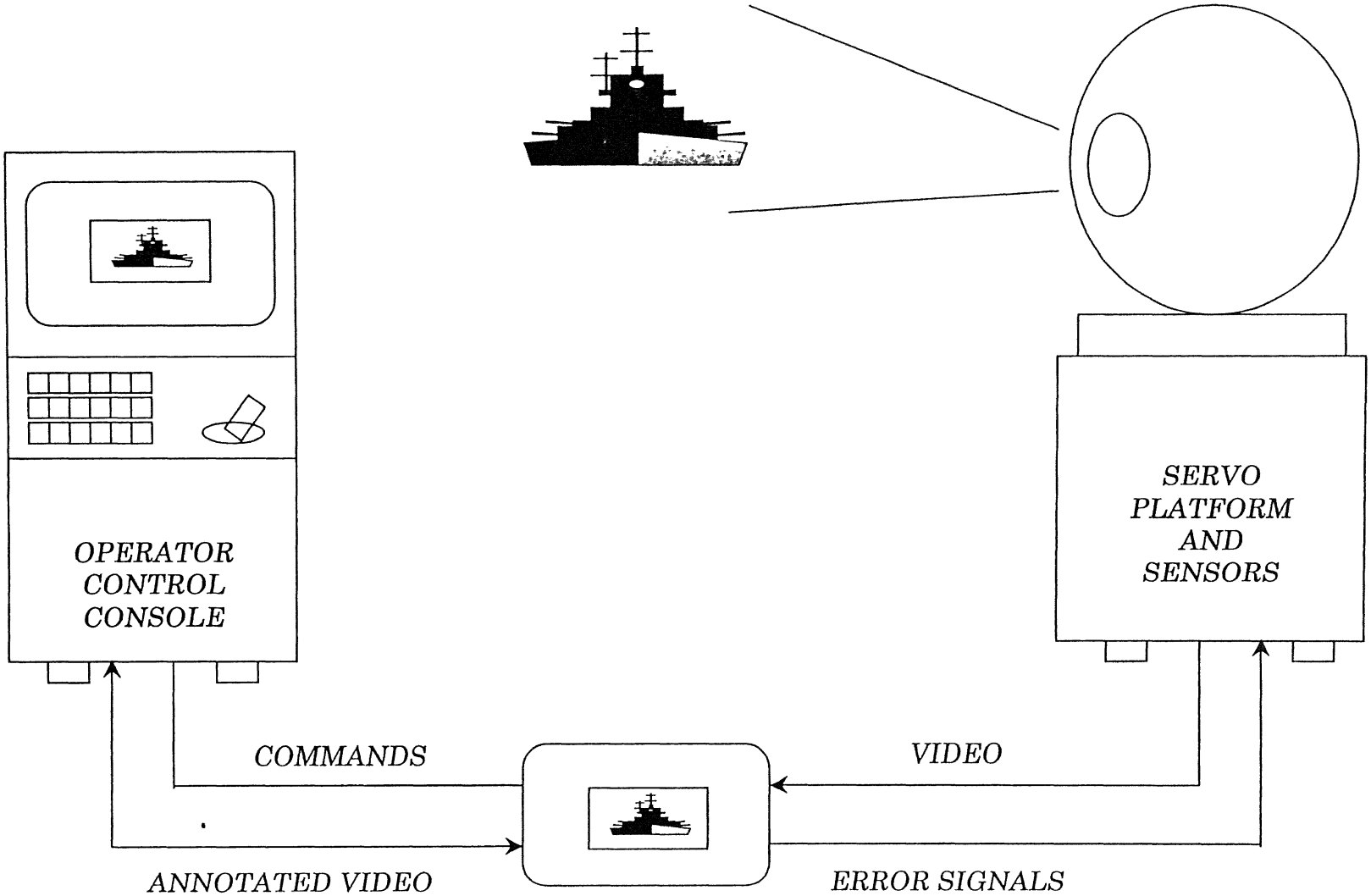


FIG 1: SYSTEM CONFIGURATION OF AUTO TV TRACKER

These positional azimuth and elevation error signals are supplied to servo system, which brings the target again in the center field-of-view. The target position has to be calculated and updated in each frame (i.e. every 20 ms) to give the real-time effect. Hence the hardware used to develop this system should be faster & algorithms should be optimized enough to cater the real time challenge.

1.1.1 TV Tracker Specifications:

The typical auto TV tracker specifications are as follows:

- Video input — *Composite video: CCIR-B std. (625 lines, 50 HZ frame rate)*
- Track gate mode — *Manual & Auto*
- Track gate position — *Center of screen (default) & Movable to any FOV position*
- Tracking algorithm — *Centroid & Correlation (Selectable to the operator)*
- Track gate Size — *Minimum 24 pixels X 24 lines*
  Maximum 64 pixels X 64 lines/ frame
  (for centroid mode tracking)
  target gate size 16 pixels X 16 lines
  Search area 64 pixels X 64 lines/ frame
  (for correlation mode tracking)
- Tracking error — *less than 1/2 pixel*
- Minimum Target Contrast — *less than 3 TV/ line = 80%*
  greater than 4 TV/ line = 25%

- Minimum Target Size — *Horizontal = 0.8 % FOV*; *Vertical = 0.4 % FOV*
- Error output — *8 bit (azimuth & elevation)*

1.1.2 Tracking Algorithms:

There are three algorithms widely used in the contrast based Auto TV tracker given as:

- Edge tracking
- Centroid tracking
- Correlation tracking

The Edge tracker seeks and acquires the first selected edge (i.e. right, left, top or bottom) of the target of interest, occurring within the gate, and maintains track on the edge. Smaller electronic gates may be used in edge tracking to gate out a major portion of the target and allow processing of only the selected track space point area. Thus, the gate of the edge tracker may be used to reject unwanted background signals. This is a significant advantage when tracking small target against a cluttered background. When the selected edge of the target occurs, horizontal and vertical position counters are sampled and the position get stored. These signals represent the position of the target edge, with respect to the internally generated reticle, or offsèt track point. These data are supplied as an output in digital form to the servo mechanism, which in turn follows the selected edge.

Two difficulties may arise in edge tracking procedure

(i). Since the differentiation is typically restricted to gated horizontal signals (i.e. differentiation of gated video in each TV lines), target that are of large extent than the track window and parallel to horizontal datum provides no track information.

(ii). Since the tracker must contain a corner within the track window

it continues to indicate an error until a corner is reached. An example of this effect is noted when the tracker locks on to a picture of a road or cloud edge and is committed to follow this edge until the track window reaches its excursion limits.

The centroid tracking is commonly used for air and naval surface targets, because these tend to be bounded targets. The centroid algorithm is so called because it finds the centroid of the target, and then designates the target aim point. The video signal from the TV camera contains a complete description of the entire field of view (FOV) of the image (target size, shape, shades of gray etc.) within the limit of the capabilities of the camera. The entire FOV of the camera appears on the TV monitor but an adjustable gate reduces the sample area of camera FOV to be used by the TV tracker. In general, the centroid mode of tracking is preferred due to its inherent immunity to noise and false alarm.

Correlation TV tracking is a process of acquiring target position information using a reference image and comparing it (each TV field) to the real time video in a defined "search window" within the FOV. The Correlation mode of tracking is used to track a target of defined shape and will maintain a track on a single feature of an extended target against a cluttered background. It maintains lock on a target, which is temporarily obscured.

The brief criteria of the selection of above discussed algorithms depends on scenario and are shown in the table 1.

CONDITIONS	*TRACKING ALGORITHMS*
Dynamic Target	Centroid
Bounded Target	Centroid
Varying background	Correlation
Target Contrast (*High*)	Centroid
Target Contrast (*Low*)	Correlation
Target Size (*Large*)	Centroid
Varying Target Shape	Centroid

1.1.3 Performance Characteristics of TV Tracker:

To accurately define the performance of TV camera, TV tracker and its servo system interface, the parameters such as range-vs-target size, target contrast and tracking bandwidth must be considered.

Range-vs-target size is most easily defined in terms of target television lines (TVL) per field as given in equation (i)

$$\text{Target TVL / field} = \frac{(\textit{target size / range}) * 57.3\ (\textit{number of TV lines / 2})}{\textit{Field of view (degrees)}} \quad \text{–(i)}$$

This equation gives the following conclusions:

- If the above equation results in TVL/ field = 1, the target is theoretically detectable by the TV systems.
- If the equation results in $2 \leq$ TVL/ field ≤ 3, the target is trackable but represents a generally poor response from the sensor.
- If the equation results in TVL/ field ≥ 4, the target is easily tracked and the sensor response is near 100 %.

In addition to target size, the contrast of the target is an important consideration. The minimum target contrast may best be explained in terms of signal-to-noise (S/N) ratio for electro-optical sensor system. The equation (ii) gives the S/N (peak-signal-to-RMS noise) for such system.

$$\text{S/ N} = \frac{(C)\ (\alpha)\ [\,MTF\ (L)]\ [\,MTF\ (S)]\ (G)}{Noise} \quad \text{— (ii)}$$

where

C = true contrast of the target to background in the spectral region of the sensor.

α = atmospheric attenuation in the spectral region of the sensor.

MTF(L) = modulation transfer function of the lens.

MTF(S) = modulation transfer function of the sensor.

G = sensor response to the signal present at the image plane.

This equation (ii) gives the following conclusions:

- If the equation results in a S/ N $\geq$ 2, the target may be tracked by a centroid tracker.
- If the equation results in a S/ N $\geq$ 4, the target may be tracked by an edge tracker.

1.1.6 Pre-processing :

Since the quality of video signal coming from the CCD or Thermal Imager heavily depends on sensor quality and the scene environment, so before the Auto TV tracker unit performs the selected tracking algorithms (i.e. Centroid or Correlation), pre-processing is required. The pre-processor makes the video signal directly useful to TV tracker. Further, pre-processor enhances track quality and incorporate some intelligence in tracker unit. The following processing is required to make the video signal useful to TV tracker:

- Pre-processing for noise suppression.
- Image processing for separation of the target from heavy clutter background.
- Image enhancement and histogram equalization (for contrast enhancement).

On completion of the above described processing the video signal will be free from noise as well as it will have acceptable contrast for TV tracker.

To provide better synchronization and ease to the operator an Auto TV tracker requires following additional circuit along with the tracking algorithms:

- Sync separator
- Threshold mode and threshold circuit
- Track Gate mode selection and Gate size

Sync Separator :

The sync separator circuit used to separate horizontal synchronization signal (i.e. H Sync, which gives the start and end of one line) and vertical synchronization signal (i.e. V Sync, which gives the start and end of frame) from the composite video signal coming through image sensor. These separated H sync & V sync information is applied to Auto TV tracker circuits for the synchronization of timing and control functions. Fig. (1.2) shows the circuit diagram for separation of sync-signal from the composite video.

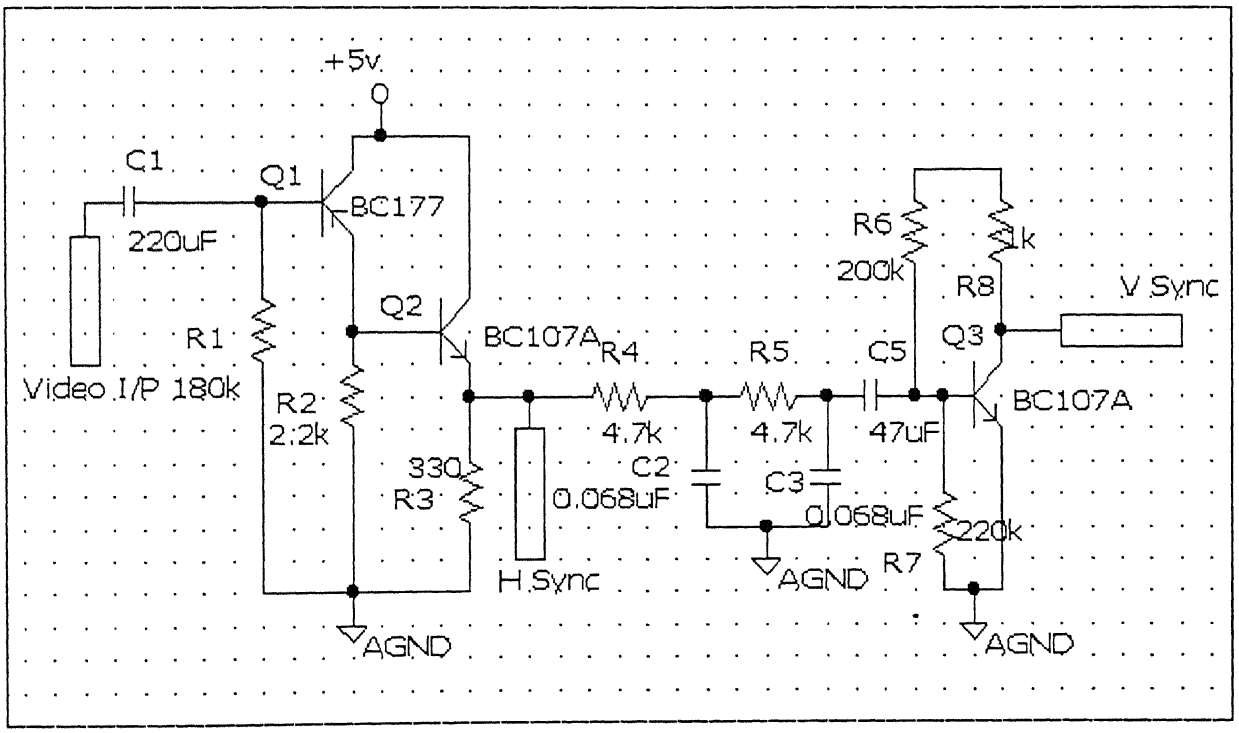


Fig. (1.2): Schematic Diagram of Snyc Separator Circuit

Threshold control:

The video signal coming form sensors are analog in nature. For the implementation of tracking algorithms (which are in digital format), video signal has to be converted in digital format. The process of converting analog video signal to binary information is called thresholding.

In this control, the operator selects a video intensity threshold, which is just below the dominant intensity of the target. The operator has to select white hot/ black hot target from analysis of the scene. It means, whether the target has the white contrast against the black background or black contrast information against the white background. Fig. (1.3) shows scheme for white hot/ black hot threshold waveform.

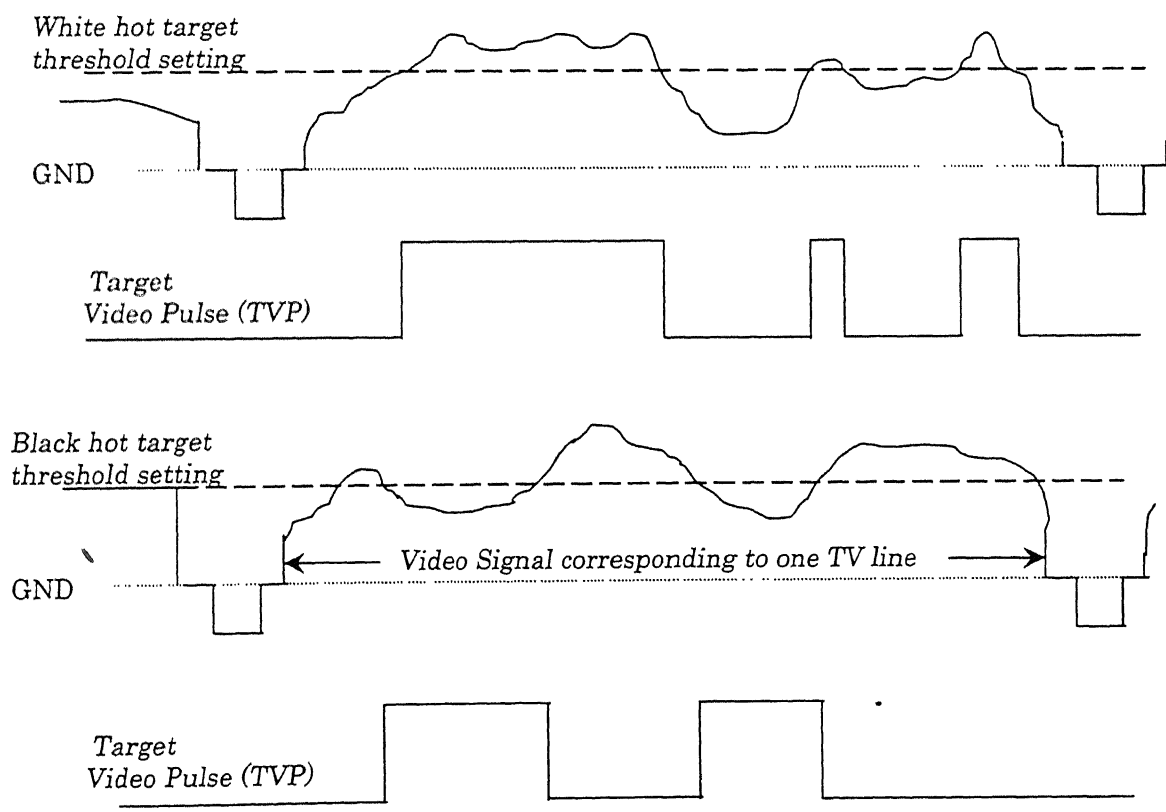


Fig. (1.3) : Waveform of Thresholding white and black hot target data

results. Chapter 3 gives the implementation and algorithmic details of the correlation tracking. Chapter 3 also describes the modules required for the implementation of correlation tracking alongwith the simulation and synthesis report. Appendix-I gives the implementation reports such as pad and map reports of the centroid and correlation algorithm implemented in respective FPGAs. Appendix-II describes the Xilinx's FPGA configuration scheme and pin details of the targeted FPGA devices used for the implementation of above proposed algorithms.

Track Gate :

Track gate is used to eliminate information other than the target of interest selected by the operator. For this purpose a track gate is positioned at the center of the screen. A switch is provided to the operator for the selection of Auto or Manual track gate size mode. This option operates in centroid mode of tracking only.

In the manual mode the operator changes the track gate size (through the UP/ DOWN switch provided in operator control console), depending upon the size of the target. This is done to ensure single target tracking.

In auto mode the track gate automatically increases or decreases in size with target variations, to maintain a constant GATE/ TARGET area relationship.

1.2 Objective of Thesis:

The objective of the thesis is to design and implement centroid & correlation based TV tracking algorithms in on FPGA for real-time application. Since the auto TV tracker is used in real-time applications, the timing constraint has to be fulfilled (i.e. the computation should be completed within one frame time). The target FPGAs for the above algorithms are XC 4008E and XC 4020E with the capacity of 8000 & 20000 gates. The maximum speed of operation of the target FPGA is 80 MHz. For the FPGA design implementation Xilinx's XACT tool has been used.

1.3 Organization of the work:

The requirements of auto TV tracker and it's algorithmic details were discussed in the present chapter. Chapter 2 describes the actual FPGA design methodology for the implementation. This chapter also describes the details of the centroid tracking algorithm to be implemented alongwith the

CHAPTER 2

Centroid Based Tracking Algorithm

2.1 Background:

The centroid based tracker is an image integrating tracker used mainly for tracking the bounded target and advantageous in the scenario where the aspect ratio of target is continuously changing e.g. Arial targets. The video signal from the TV camera (CCD or Thermal Imager) contains the complete description of the entire field-of-view (FOV). The entire FOV of the camera appears on the monitor but an adjustable track gate reduces the sample area of the FOV to be used by the tracker to compute the error signal in azimuth and elevation direction. Centroid based tracker integrates the video within the track gate by partitioning the active area into a matrix as shown in fig (2.1).

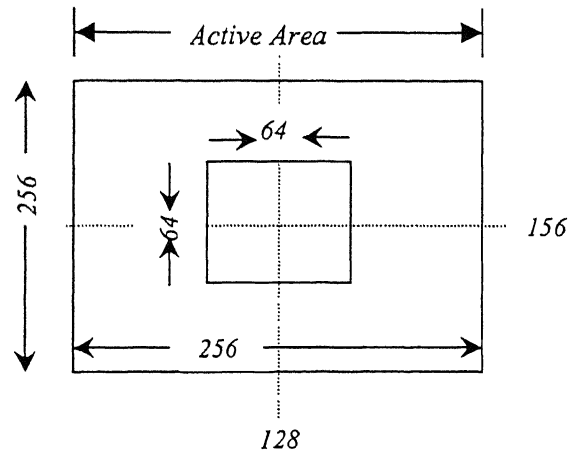


Fig.(2.1): Scheme of Partitioning the active area

The Horizontal sync in the CCIR-B standard video format has the period of 15.625 KHz. So if 4 MHz clock is chosen for scanning then the active area can be

defined in terms of (4 MHz/ 15.625 KHz) = 256 pixels. The track gate size is shown in fig. (2.1) is for 64 pixels/ 64 lines.

For computation of centroid of the target within the track gate, the video signal is digitized using threshold scheme to get the binary image as shown in fig. (2.2).

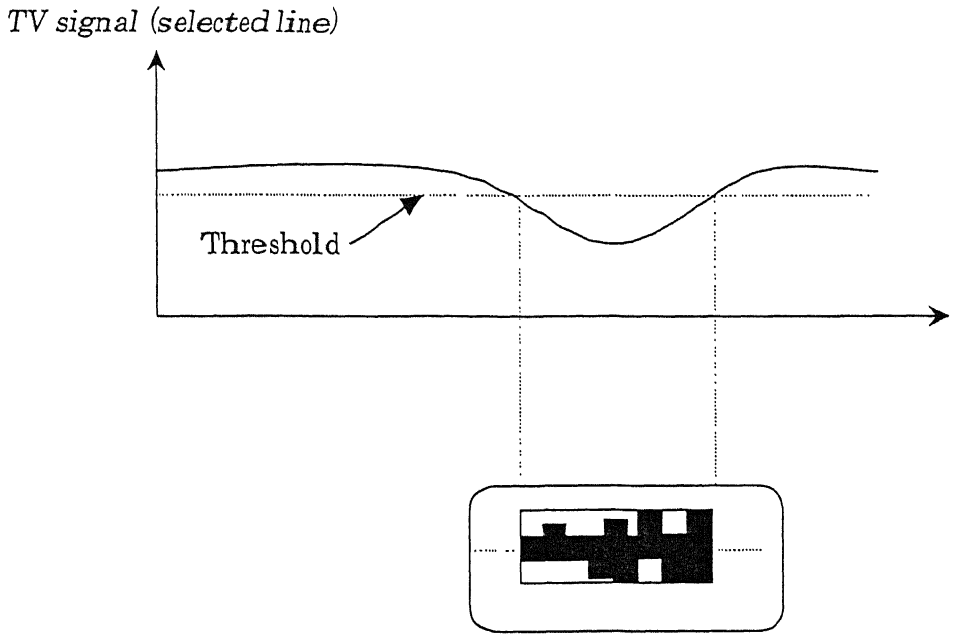


Fig. (2.2) : TV picture (binary target representation)

The above figure shows the binary target representation of the video signal corresponding to one line. The video information is digitized in successive lines to generate target video pulses (TVP). The tracker integrates the horizontal and vertical coordinate data of the digitized video signal to compute the centroid of the target within the track gate using the equation (i) and (ii) given below:

$$X_{CENTROID} = \frac{\sum_{i=0}^{m} X_i}{\sum_{j=0}^{n} N_j} \qquad \text{------ (i)}$$

$$Y_{CENTROID} = \frac{\sum_{i=0}^{m} Y_i}{\sum_{j=0}^{n} N_j} \quad \text{----- (ii)}$$

Where

N = data point

X_i = X- position of the target

Y_i = Y- position of the target

n = total number of data points

m = total number of position coordinates

These centroid signal are use to compute the error signal with respect to the center of screen and being send to servo system, which will bring the target again in the center of the monitor in next frame.

2.2 FPGA Based System Design:

A typical flow diagram for the FPGA based system design methodology has been shown in the fig. (2.3). The design starts with system specification, which is converted into the modules and sub-modules. These modules are initially designed, synthesized and simulated functionally using Hardware Description Language (e.g. VHDL). These modules after the simulations are interconnected according to the specification. After the interconnection, the complete design is synthesized and simulated for the particular FPGA device. The gate level synthesis gives the actual gate density required for the design. After the layout synthesis a bit file is generated for the target FPGA. This bit file is used for the configuration of on board FPGA device.

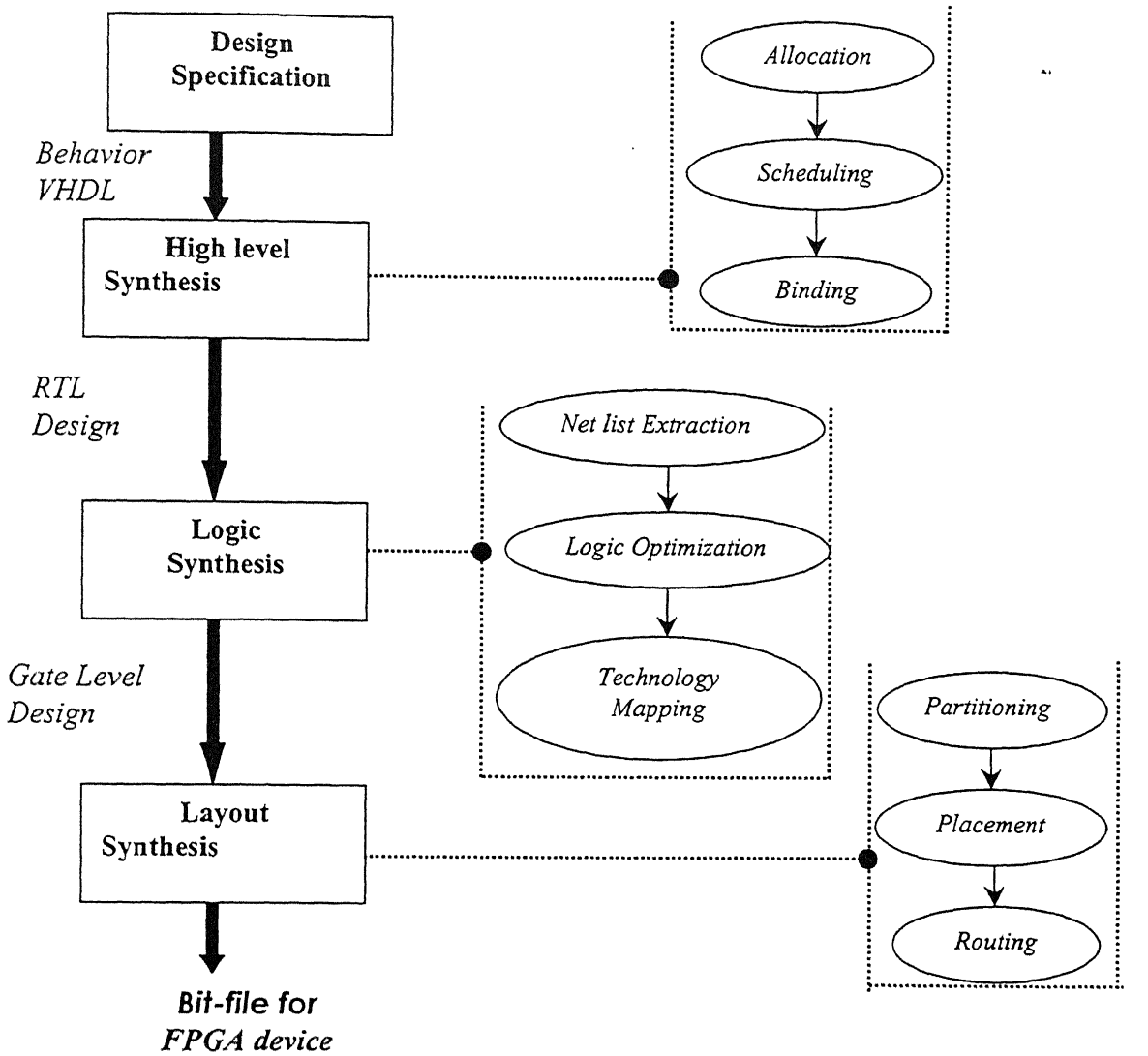


Fig. (2.3) : Flow diagram for FPGA based design methodology

2.3 Design of Centroid Based Tracking Algorithm:

For actual implementation of the centroid algorithm, scheme is shown in fig. (2.4). The video signal is first digitized using the video threshold circuit with the help of reference voltage so that the target contrast level should be above the threshold level. This will generate the binary image of the target of interest. To reject the information other than the target, the track gate is generated with the two selectable modes. One is Manual mode and second is auto gate mode. The operator can select either of these modes through control switch provided on

operator control console. The track gate alongwith binary information is processed to generate target video pulse (TVP). Basically TVP generator is AND gate which give the binary information within the track gate. The TVP within track gate goes to the computational logic block, which computes centroid of the target using eq. (i) & (ii) with horizontal and vertical sync as a control signal.

All logic except the threshold is digital in nature. In the present work, implementation of computational logic is modeled in the VHDL and finally fused in the Xilinx FPGA XC 4008E.

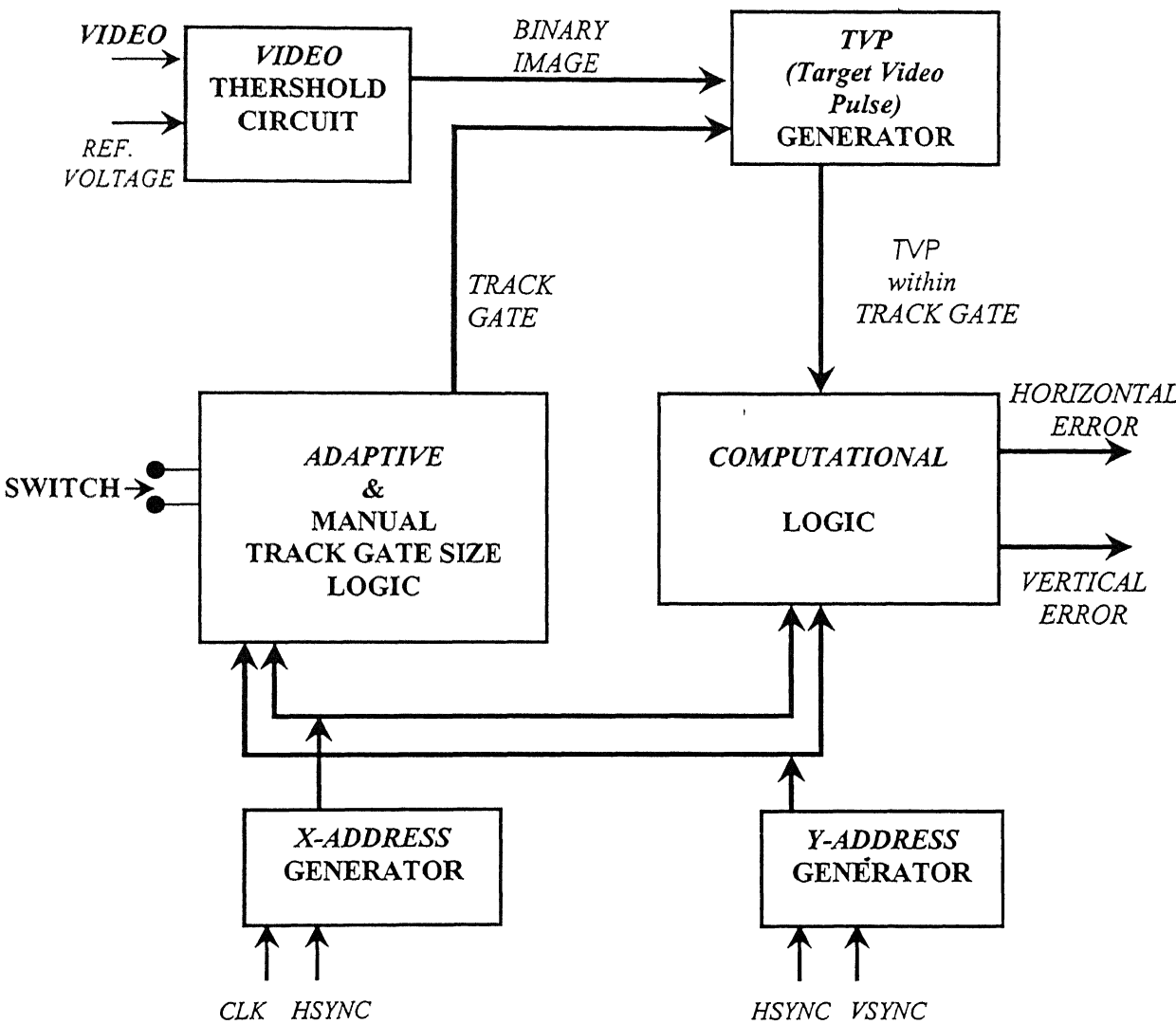


Fig.(2.4): Scheme of Centroid Tracking Implementation

2.3.1 Generation of Manual Track Gate:

The scheme for the generation of manual track gate size is shown in fig. (2.5).

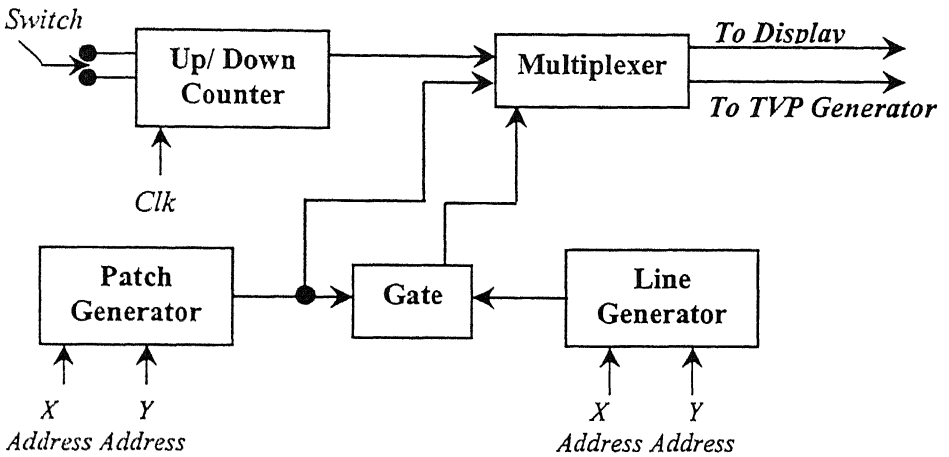


Fig. (2.5): Scheme for Manual Track Gate Size

In the manual mode track gate size the operator has the flexibility to increase or decrease the track gate size depending upon the scenario, within eight steps from maximum (64 pixels X 64 lines) to minimum (24 pixels X 24 lines). For this purpose a switch is provided. The patch of above sizes and corresponding lines are generated using horizontal and vertical address generator in separate modules in VHDL. These lines and patches are passed through the AND gate. These gated lines are used for display purpose, which gives the visual effect to operator (i.e. the target of interest is confined to the area covered by line pattern). The output of the AND gate alongwith the patch generated goes to the multiplexer. The multiplexer output ensures that the same patch pattern (to be used as a control signals by the computational block & TVP generator) alongwith line pattern goes to the display.

2.3.2 Generation of Automatic Track Gate:

The scheme for the generation of automatic track gate size is shown in fig. (2.6).

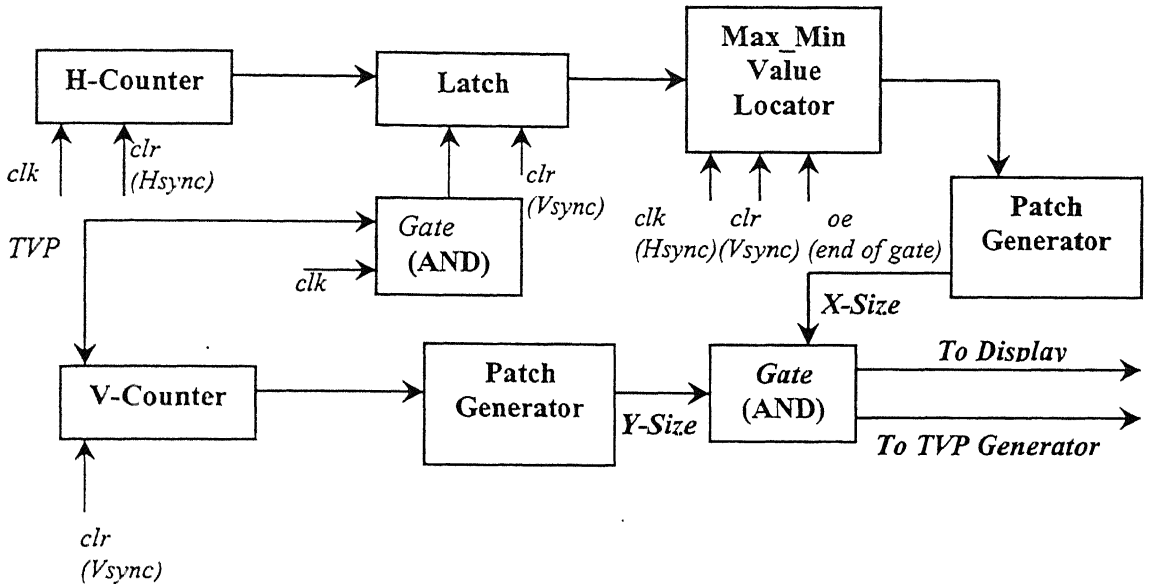


Fig. (2.6): Scheme for Auto Track Gate Size

In auto track gate size mode, the track gate automatically increases or decreases in size with target variations, to maintain a constant GATE/ TARGET area relationship. This will lead to single target tracking. In this mode the initial track gate size will have its default value and in the next frame onwards it will confined the as per the target size. The H-counter will generate the address using 4 MHz clock and h-sync. The TVP inside the clock is used as a clock to the latch, which will store the corresponding address of the target. The Max_Min value locator is used to store the maximum and minimum value of the address latched. Those will be the lower and upper limits of the target size. The patch generator will generate the patch in the X-direction using these limits. For the Y-direction the vertical counter will count the no. of TVP in a frame and will generate the patch. Finally these patches are used to generate the control signal for computational logic and lines for the display.

2.3.3 Computational Logic:

The calculation of the centroid of the target is computationally intensive. Since the tracking is a real-time operation hence the hardware should be fast enough and logic should be optimized to cater the requirement. The implementation scheme of computational logic is shown in fig. (2.7).

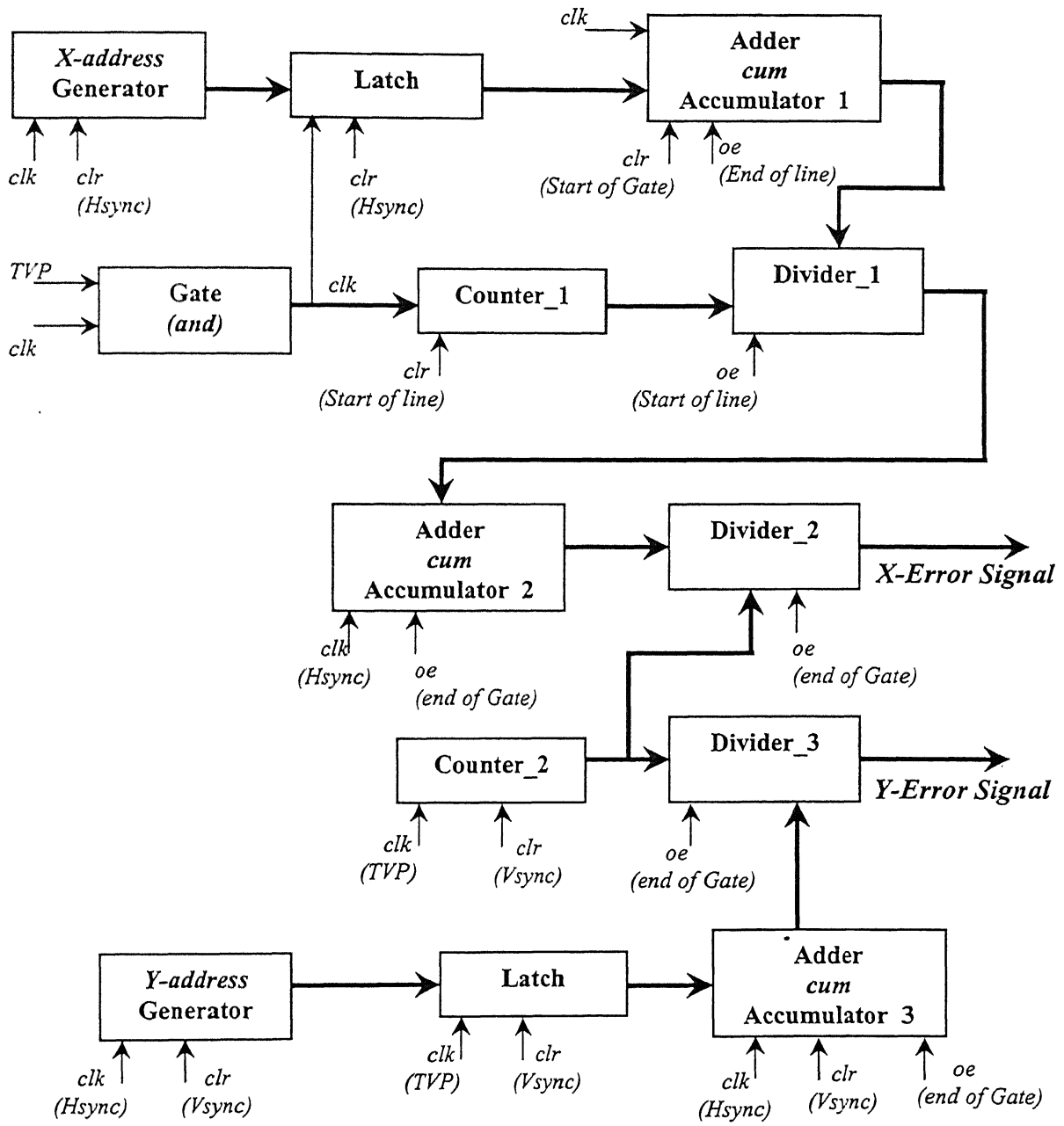


Fig. (2.7): Scheme for Implementation of Computational Logic

The X- address generator will give the continuous address as per the partitioning scheme suggested in fig. (2.1). The latch will store the corresponding address of the target using the clocked target video pulse (TVP) as a clock. These latched addresses will get added and accumulated for a single line. At the end of the track gate line, the accumulated value will be applied to the divider_1 logic block, which is the dividend. At the same time the counter_1 block will count the number of clock pulses inside the TVP and fed to the divider_1, which is the divisor. The divider_1 block gives the quotient as an integer. The quotient will be fed to the accumulator_2 with the rising edge of the h-sync. This operation will continue for each line upto the last line of the track gate. Finally the accumulator_2 value will be divided by the number of TVP present in one frame. This will be the $X_{CENTROID}$ or the horizontal error signal of the target.

For the calculation of the $Y_{CENTROID}$ or the vertical error signal of the target, the Y-address generator will give the vertical address using h-sync as a clock and v-sync as a clear signal. The latch will store the corresponding address with the every rising edge of the TVP. The accumulator_3 will add & accumulate all the latched addresses and fed to the divider_3 block, which will divide the accumulated value by the number of TVP counted by the counter_3. This is the Y-error signal of the target of interest.

Fig. (2.8) shows the waveform of the control signal such as target video pulse (TVP), clocked TVP (i.e. the clock inside the TVP) etc. This figure also shows the complexity involved in the calculation of the centroid of the target. For the case of 64 X 64 track gate size, time left for the calculation is 24 μsec. for each line. So the conventional discrete IC's can't do the calculation that fast, but the FPGA (with 1-3ns delay) can solve the purpose. The key component in this design is the divider logic, which takes most of the time. The design of the divider logic is discussed in the next section. To avoid any timing conflict the control signal is modified to carry the accumulated value till the start of next track gate line. That is ***loop pipelining*** is used, which will increase the division time from 24 μsec to (24 μsec + 24 μsec) = 48 μsec.

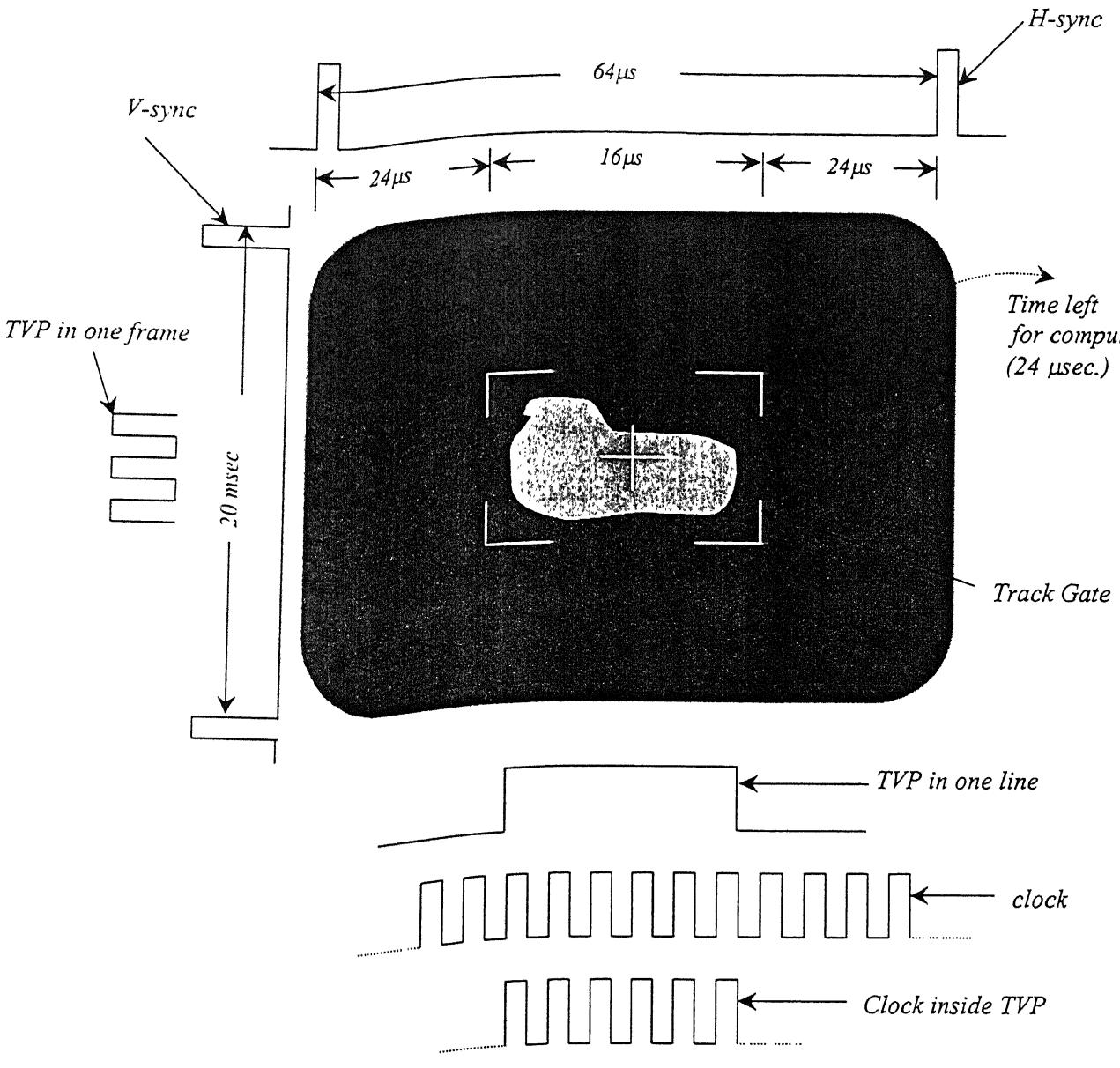


Fig. (2.8): Waveform of Control signals used in the Computational Logic

Design of divider :

The logic used for the design of divider is shown in fig. (2.9). The components used for this logic are comparator & substractor, multiplexer and counter. For the first clock cycle MUX will send the input A (i.e. dividend) and it will get compare with the input B (i.e. divisor).

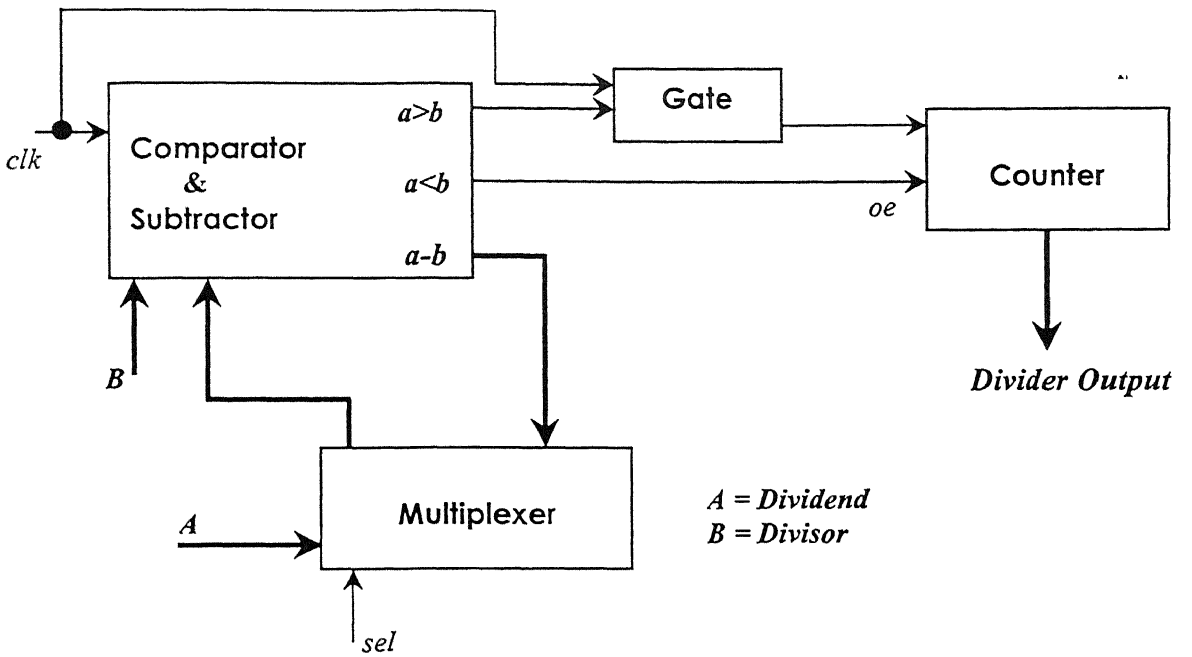


Fig. (2.9): Scheme for Divider Logic

If this value is less than A, then this block will send the subtracted value to MUX as its second input. Simultaneously the a>b will be high. In second clock onwards the MUX will send its second input and similar operation will be repeated till the a<b is high. As long as the output a>b is high, the clock signal gated with a>b serves as a clock to counter. The counter counts the no. of clock pulses and this will be the output of divider (i.e. quotient). The divider shows its output only when the a<b is high (i.e. output enable).

2.4 Results and Discussion:

The divider logic have been simulated and synthesized first to check the requirement. The figure (2.10) shows the simulation result of the divider logic. Since the designed divider is clock cycle dependent, hence it can be best analyzed by the worst-case inputs. For the case of 64 X 64 track gate size, the starting address of the line is 96 and ends at 160 (according to the partitioning

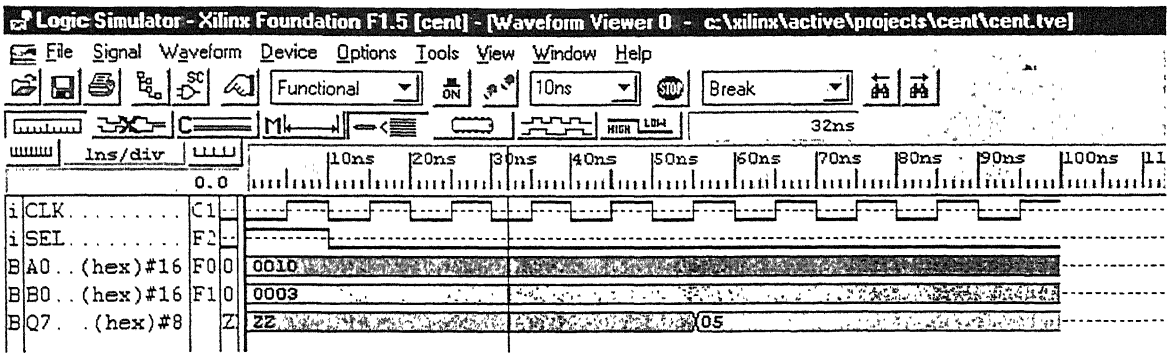


Fig.(2.10) : Simulation result of the divider implemented

scheme suggested above). So if all the pixels are active i.e. the whole line of the track gate is covered by the target then the divider requires the 8320/ 64 = 130 clock cycle to complete the division operation. With the clock of 4 MHz it will take 32.5 μsec, but using loop pipelining the available time is 48 μsec. For the worst case, when only the last pixel of the track gate line i.e. 160th pixel is active, then the divider will require the 160/1 = 160 clock cycle to complete the division operation. It means it will take 40 μsec. So the 8 μsec time is left to cater the interconnect delay. Fig. (2.7) shows the divider output $(05)_H$ with the inputs $(0010)_H$ and $(0003)_H$.

The fig. (2.11) shows the simulation results of the computational logic implemented to calculate the centroid of the target. The target video pulse (TVP) is been detected at the addresses 03, 04 and 05, which are added and accumulated. The counter counts the no. of clock cycle inside the TVP i.e. 03. The divider gives the final X-error as 04. Similarly for the Y-error the vertical address 04, 05 and 06 are added and divided by the no. of the TVP in a frame i.e. 03. So the Y-error is 05 as shown in the simulation result.

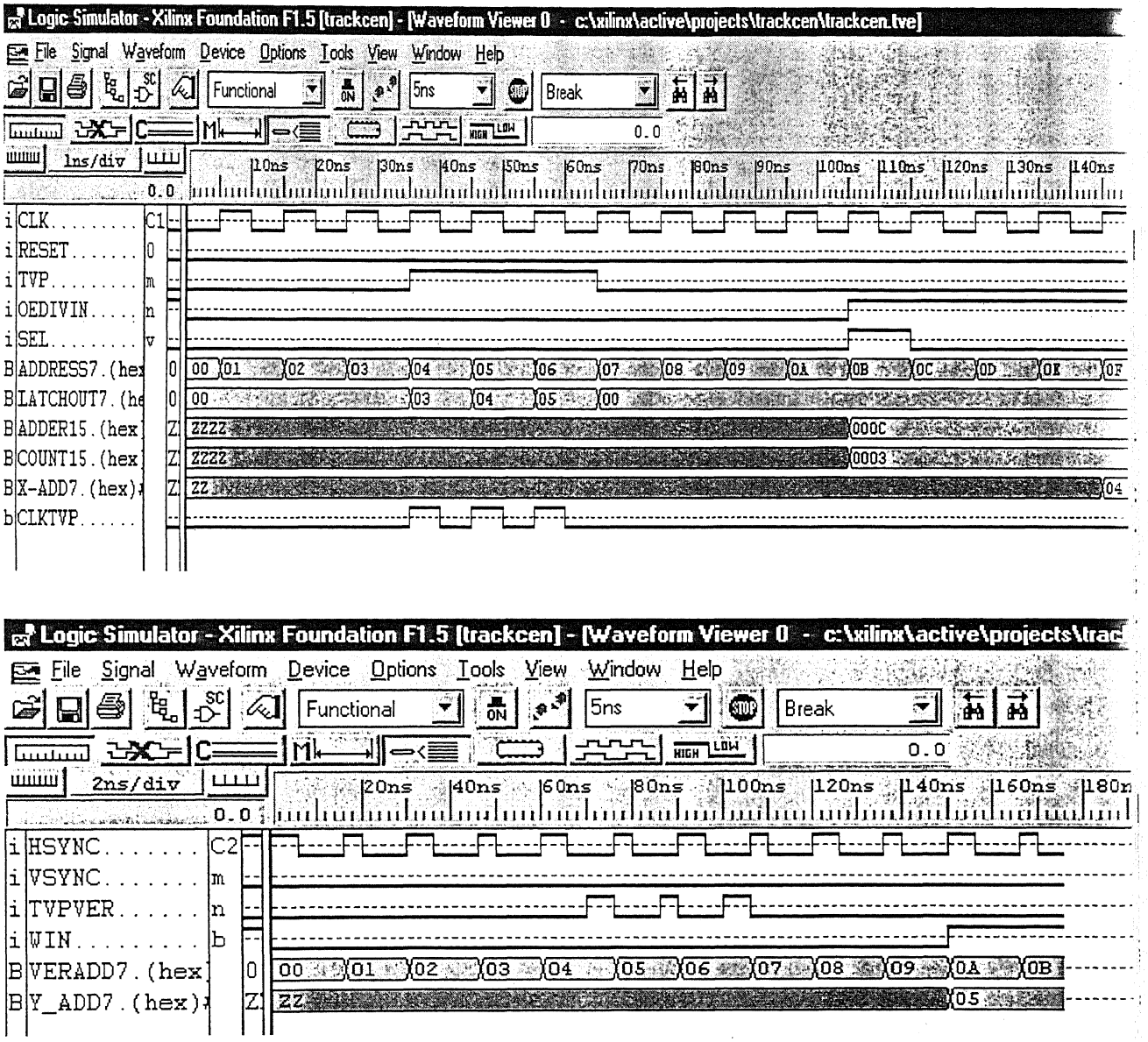


Fig.(2.11) : Simulation result of the Computational logic implemented

For the worst case, divider requires 160 clock cycles. Since the available time for the division is 48 μsec. The required operating frequency will be 48/160 = 0.3μsec = 3.3 MHz. It means that the system can use the clock frequency of 4MHz. The point to be noted that if the loop pipelining is not being used then the minimum required frequency would have been 6.6 MHz, as the available time for the division reduces to 24 μsec.

The comprehensive synthesis and implementation result of the centroid tracking algorithm into FPGA XC4008E has given below:

Number of CLBs	: 233 out of 324
CLB Flip Flops	: 190
4 input LUTs	: 115
3 input LUTs	: 54 (15 used as route-through)
Number of bonded IOBs	: 25 out of 61
IOB Flops	: 0
IOB Latches	: 0
Total equivalent gate count for design	: 6425
Additional JTAG gate count for IOBs	: 1200

The detailed map & pad reports are attached in the appendix-I.

CHAPTER 3

Correlation Based Tracking Algorithm

3.1 Background:

Finding the location of a reference image within the field of view (FOV) of a video sensor in real time is considered using a correlation technique. Real time means that the reference image location within the live video image FOV is found for each field of the video sensor. This requirement prevents not only a buildup of data requiring large memory but also does place severe speed requirements on the algorithm chosen. For a standard image sensor (CCD or thermal imager) with a frame rate of 25 Hz, 2:1 interlace, and corresponding field rate of 50Hz, each match point must be computed in 1/50 Hz = 20ms. The correlation tracking is an effective method of targets that do not have the salient characteristics. The correlator essentially computes a surface or map of coefficient, which are a measure of similarity of the reference to all portion of the scene. The location of the highest match is used to update the aim-point of the sensor.

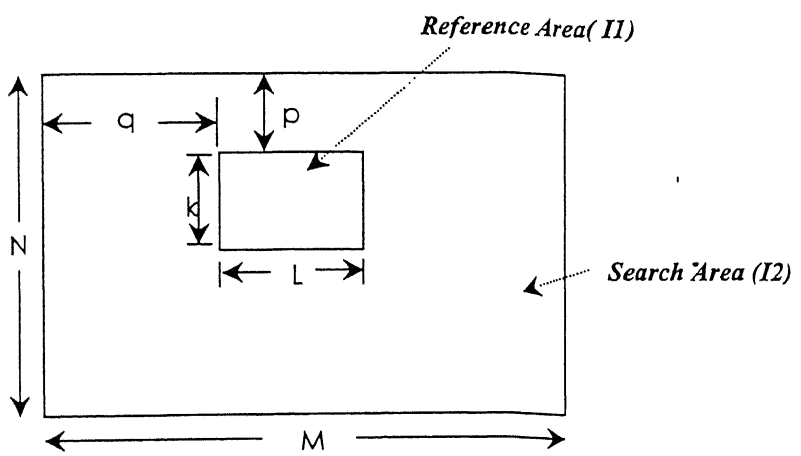


Fig. (3.1) : Geometry of correlating reference image with search image

Figure (3.1) shows the geometry of correlating K x L reference image (I1) with N x M sensor live image (I2). Both the images are digitized and quantized into pixels. Additional preprocessing of one or both images will be necessary if the assumptions described in the chapter1 are not met.

Two algorithms are known for the implementation of correlation tracking technique: one which is the discrete version of the classical definition of correlation, and other which is related to this classical definition.

The classical approach to the problem of determining where two signals match is correlation. Consider two functions $f_1(t)$ and $f_2(t)$, the correlation integral can be defined as equation (i)

$$C(\gamma) = \int_{-\infty}^{\infty} f_1(t) f_2(t+\gamma) dt \quad \text{---------- eq. (i)}$$

where γ is allowed to take the values $-\infty$ to $+\infty$. The value of γ maximizes $C(\gamma)$ in equation (i) is the correlation peak and is defined to be the match point between the two signals. It is obvious that determining the correlation peak consists of multiplying one signal by the other signal, shifted by γ, and then evaluating the area under resulting curve.

If the both signals are sampled and held, then equation (i) can be approximated by the following equation:

$$C(p) = T \sum_{n=-\infty}^{\infty} f_1(n) f_2(n+p) \quad \text{----------eq. (ii)}$$

Where $f_1(n) = f_1(nT)$ *and* $f_2(n+p) = f_2(nT+pT)$ and T is the sampling interval. As T becomes very small, the result in eq. (ii) approaches that of eq. (i) where the p which maximizes C(p) is defined as the correlation point.

The digitized video image (I2) is represented by an N x M array of pixels as shown in fig.(3.1). The values of N and M are fixed by the choice of sample rate and number of TV lines per frame. For our design it is 64 x 64. The reference image (I1) is represented by a K x L array of pixels. For our design it is 16 x 16. The p and q dimensions in fig. (3.1) give the vertical and horizontal position,

respectively, of the reference in the video image. These indices start in the upper left corner of I2, where p = q =0. One possible two-dimensional discrete correlation algorithms (often referred as the direct method) is given by

$$R(p,q) = (1/KL)\sum_{n=1}^{K}\sum_{m=1}^{L} I1(n,m)I2(n+p,m+q) \quad \text{---------- eq. (iii)}$$

for $0 \le p \le N\text{-}K,\ 0 \le q \le M\text{-}L$

where R(p, q) is the correlation function, and the division by KL is the scaling factor. In eq. (iii) for each possible value of p and q, R(p, q) is obtained by summing the product of the KL corresponding pixels in the two images. The combination of p and q (denoted by p' and q'), which maximizes R(p, q)is the image registration or match point.

Another algorithms for determining the similarity or the amount of registration between two images is known as a *sequential similarity detection algorithm* (SSDA) . The SSDA registration surface is computed by adding the KL absolute values of the difference between corresponding I1 and I2 pixels for each value of p and q as given by equation (iv).

$$E(p,q) = \sum_{n=1}^{K}\sum_{m=1}^{l} | I1(n,m) - I2(n+p,m+q) | \quad \text{------- eq. (iv)}$$

for $0 \le p \le N\text{-}K,\ 0 \le q \le M\text{-}L$

The registration point (denoted by p* and q*), is the value of p and q for which E is minimum.

For the implementation of correlation tracking technique in this work, SSDA method has been chosen because of the following reasons:

- The SSDA method implementation requires less hardware since it uses subtraction rather than multiplication (used in direct method). Hence the implementation of direct method would require the tracker generally to have extensive hardware/ software capability and normally cannot be

performed in real-time, which is a basic requirement for electro-optical tracking systems.

- Since the SSDA method uses the absolute difference of the search and reference data, hence it can perfectly register the white target against the black background as well as the black target against the white background. But the direct method cannot have this flexibility. The operator doesn't have to change the polarity of the target in correlation mode of tracking, when SSDA method is used.

3.2 System Configuration of Correlation Tracking Algorithm:

The scheme of the system configuration is shown in fig. (3.2). The tracker accepts the video signal from image sensor either CCD or Thermal Imager.

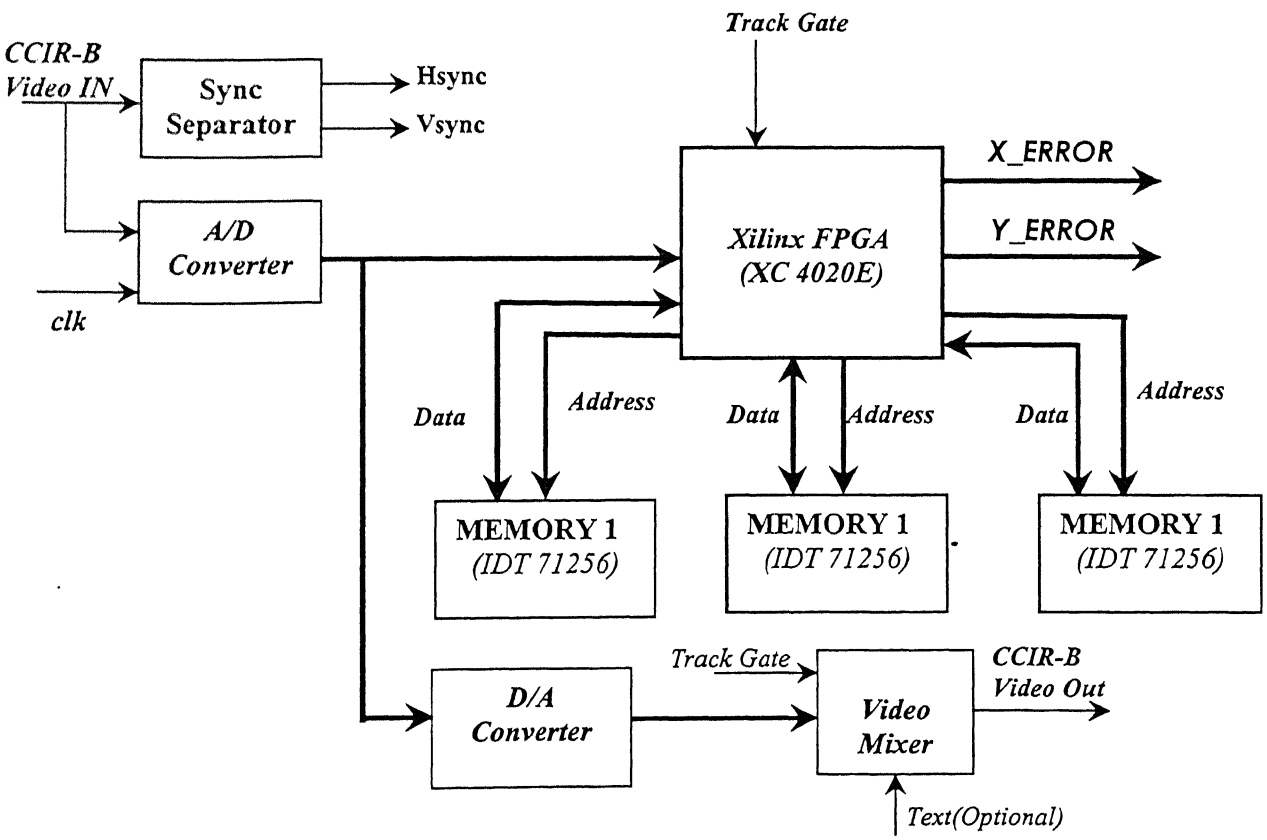


Fig. (3.2): System Configuration of Correlation based Tracking

The sync separator gives the h-sync and v-sync from the video signal received, as per circuit shown in fig . (1.2). The A/ D converter (using AD9048) digitizes the analog video information of the scene into 8-bit data format. These ADC data are used in the computation of match point. In this system three memory approach has been used to cater the real-time requirement. The live image (64 x 64) will be stored first in the memory1 and the next frame will be stored in memory 2(64 x 64). At the start of the third frame the data form the center of the first memory (16 X 16 size) will be read and it is the reference data. The memory 2 will be treated as the search data. Simultaneous the memory 3 will store the third frame. Computation logic (SSDA) will compute the best match point and send the error signals to the servo system, which will bring the reference image portion again to the center of FOV. In the fourth frame the memory 2 will give the reference data, memory 3 will give the search data (track data), memory 1 will be in the writing mode and vice-versa. This scheme has a latency of one frame, but computes the position of the target in a one frame (i.e. in real-time).

The computational logic, address scheme for the reference & search memory and control logic have been designed in VHDL and implemented in FPGA (XC4020E).

3.3 FPGA implementation of Correlation Based Tracking:

Fig. (3.3) shows the scheme of implementation of correlation tracking algorithm using FPGA. The design has been done by using the various modules such as main controller, address generator, interconnect logic and computational logic.

Main Controller: The main controller ensures that the operations with respect to the three memories should change accordingly with every three-frame pulse.

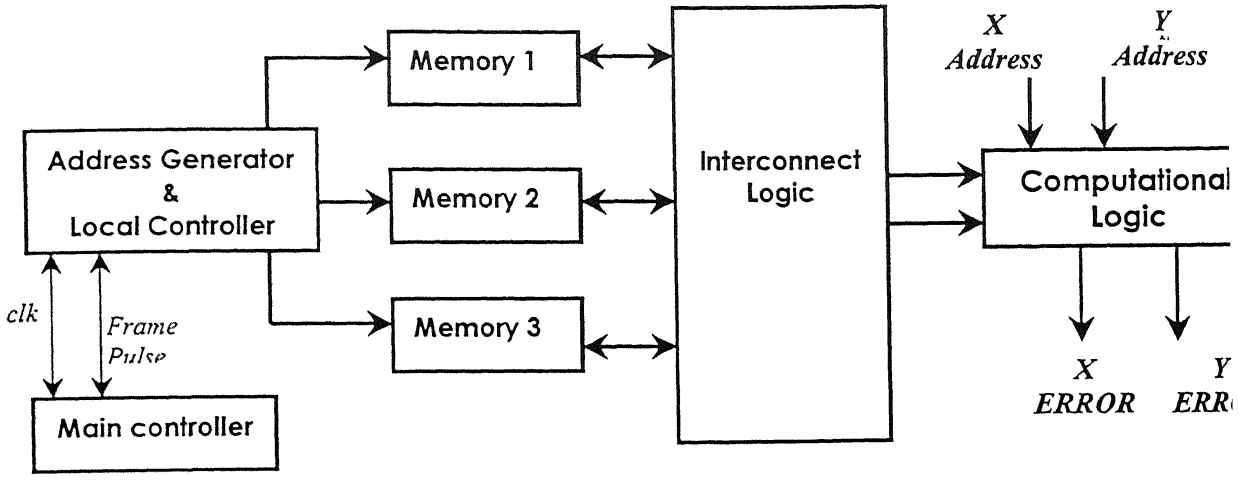


Fig. (3.3): FPGA Implementation of Correlation Tracking Algorithm

Address Generator: The address generator gives the writing and reading address to the respective memories as per the scheme described in section 3.2. The local controller ensures that the Read/ Write signal must go along with the respective addresses. The result of the address generator has been discussed in the next section.

Interconnect Logic: The interconnect logic ensures that there should not be any bus data conflicts i.e. in INPUT data or OUPUT data of the memories. Since these memories either will be in writing or reading mode (out of which two are in reading mode & one in writing mode at a time). The control signal of the memories will change accordingly. The fig. (3.4) shows the logic scheme designed in VHDL. The control signals C_{11}, C_{12} & C_{13} are the signals used by the memory 1 as chip select ($\overline{CS}$), output enable ($\overline{OE}$) & input/ output (I/ O) respectably. Similarly C_{21}, C_{22} & C_{23} are the control signals for the memory 2 and C_{31}, C_{32} & C_{33} are the control signals for the memory 3. These control signals have been generated by using finite state machine (FSM) as per the scheme shown in fig. (3.5). The integration of above two schemes gives the complete design of interconnect logic.

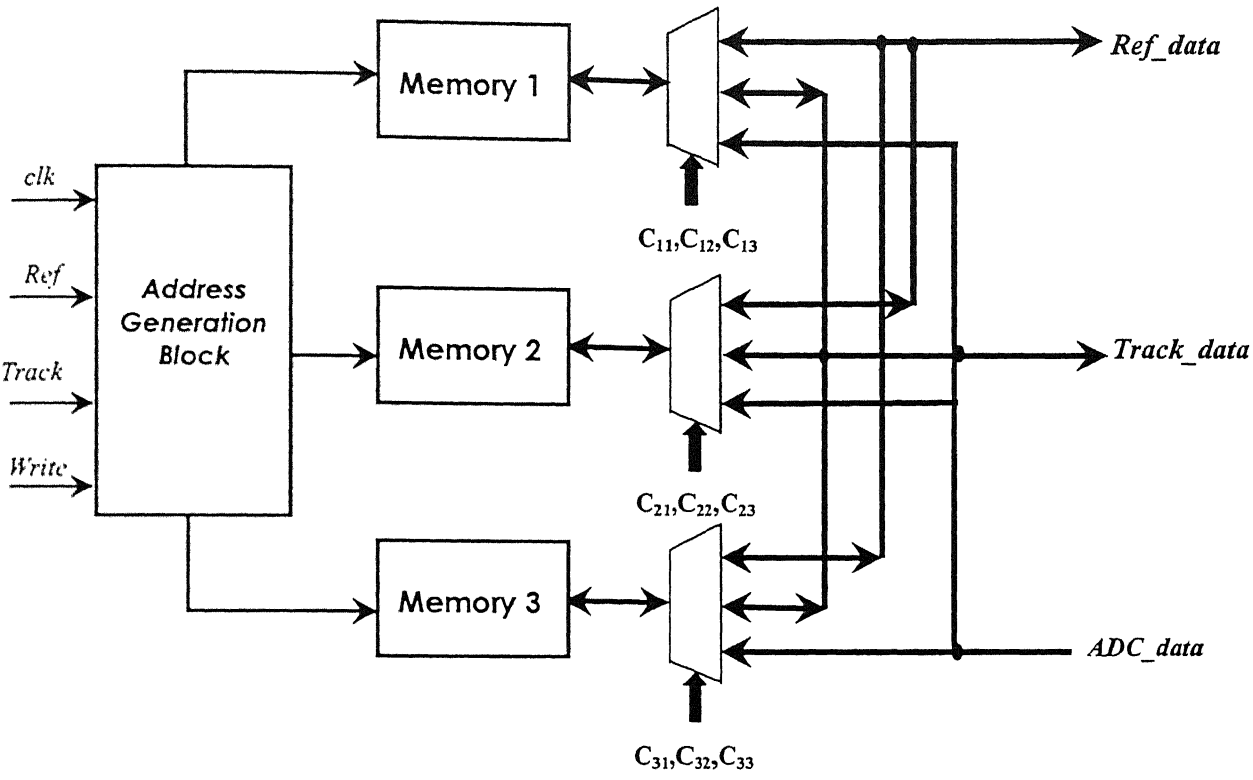


Fig. (3.4): Interconnect Logic for Correlation Tracker

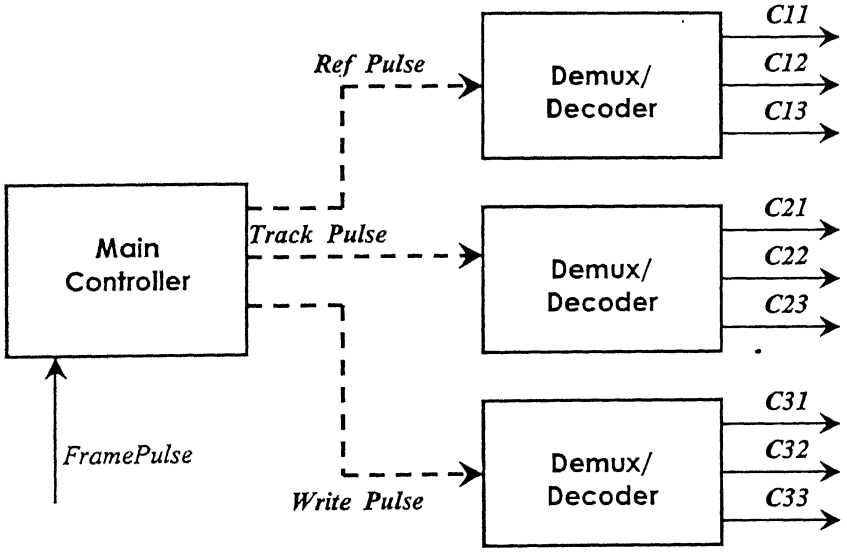


Fig. (3.5): scheme for the implementation of memory Controller

Computational logic: The computation logic uses the sequential similarity detection algorithm (SSDA) for the calculation of the image registration point. It computes the absolute difference of the reference data & track data (or search data) for the 16 x 16 = 256 pixels i.e. the one location. Likewise it will compute the value for all the search position.

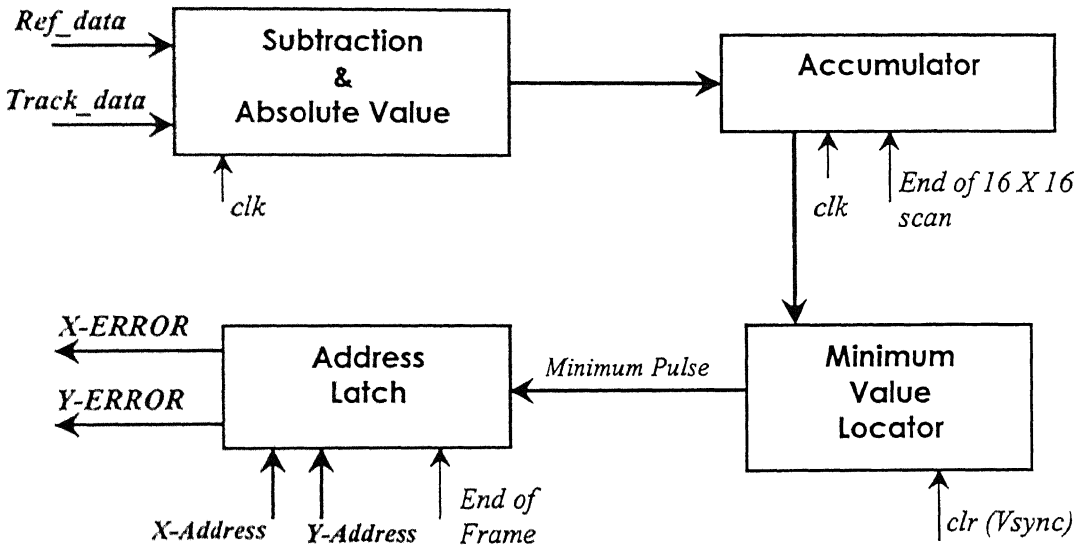


Fig. (3.6) : Scheme of Computational Logic for Correlation Tracking

The fig. (3.6) shows the implementation scheme of computational logic. First it will compute the subtracted and absolute value of one pixel of reference data with track data, likewise it will compute for the whole 16 x 16 image size. Then the accumulated value will go to the minimum value locator, which will store the minimum value among the all search area. It will generate the minimum pulse whenever it gets the new minimum value. This minimum pulse uses as a clock to latch the X & Y address at the minimum value. This will be the X-error & Y-error signal.

3.4 Results and Discussion: The simulation results of address generator for the correlation tracking implementation are shown in the fig. (3.7). The memory 2 is in the reading mode and gives the reference data from the center of the image size 64 x 64. So it starts with $(1818)_H$ and goes up to $(1827)_H$. Then it start with $(1819)_H$, till it reaches the address $(2727)_H$ which complete the 16 x 16 block as shown in fig. (3.8).

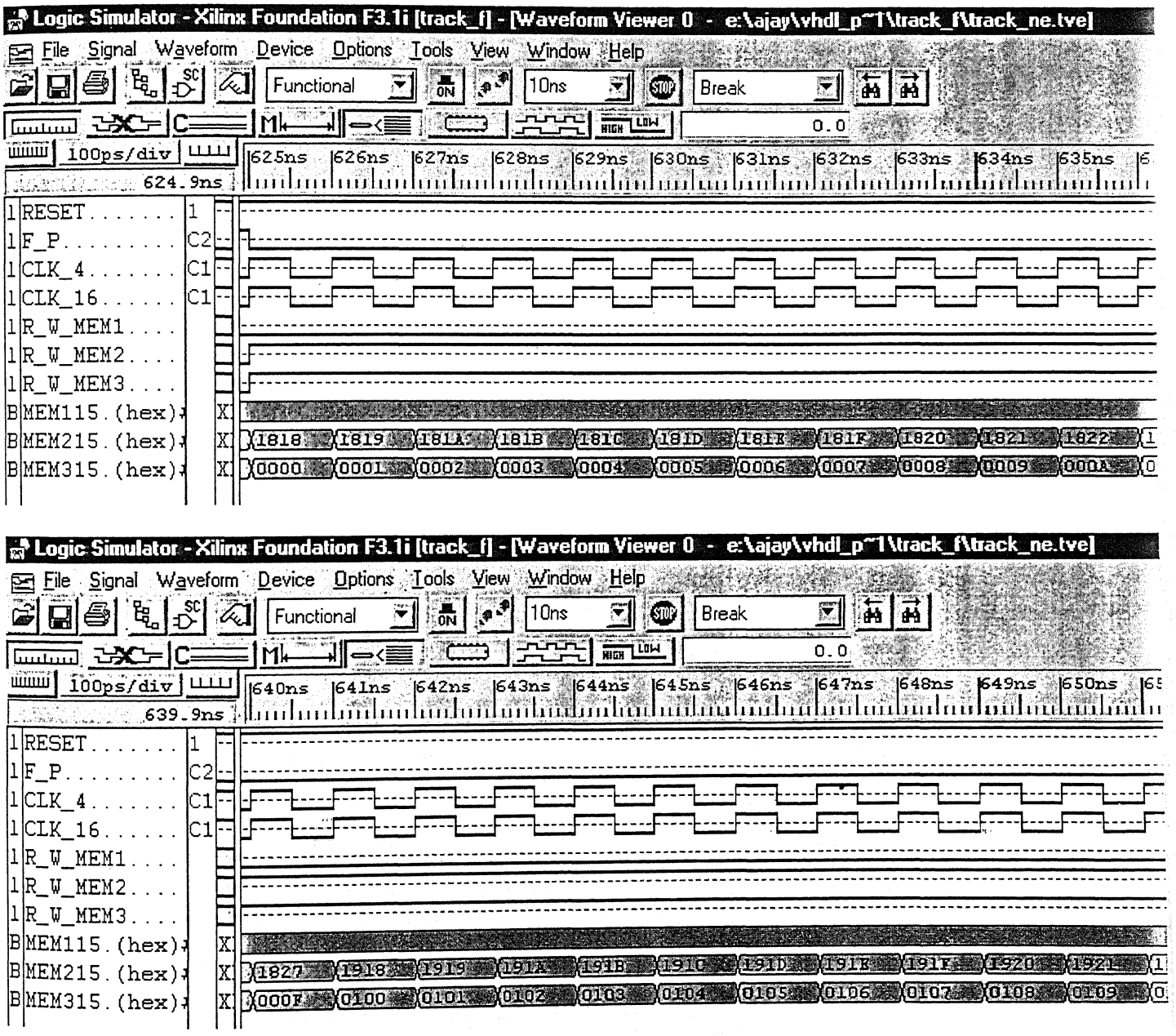


Fig. (3.7): Simulation result of the address Generator

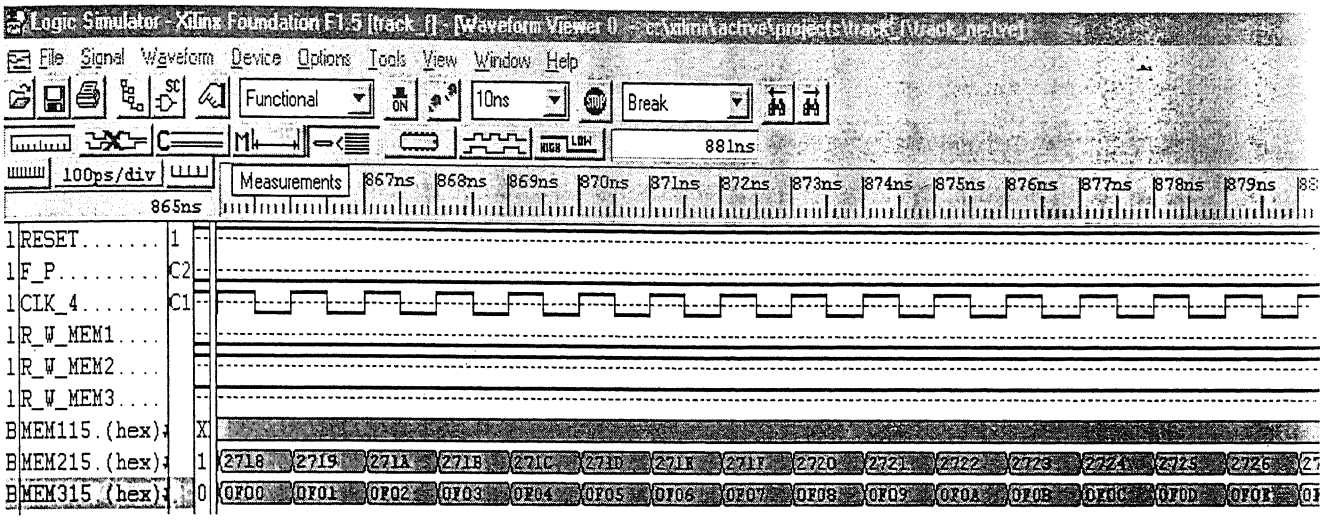


Fig. (3.8): Simulation result of the address Generator shows the completion of 16 x16 block

Fig. (3.9) shows the simulation result of the interconnect logic. This shows the various control signals used by the three memories are as per the detail described above.

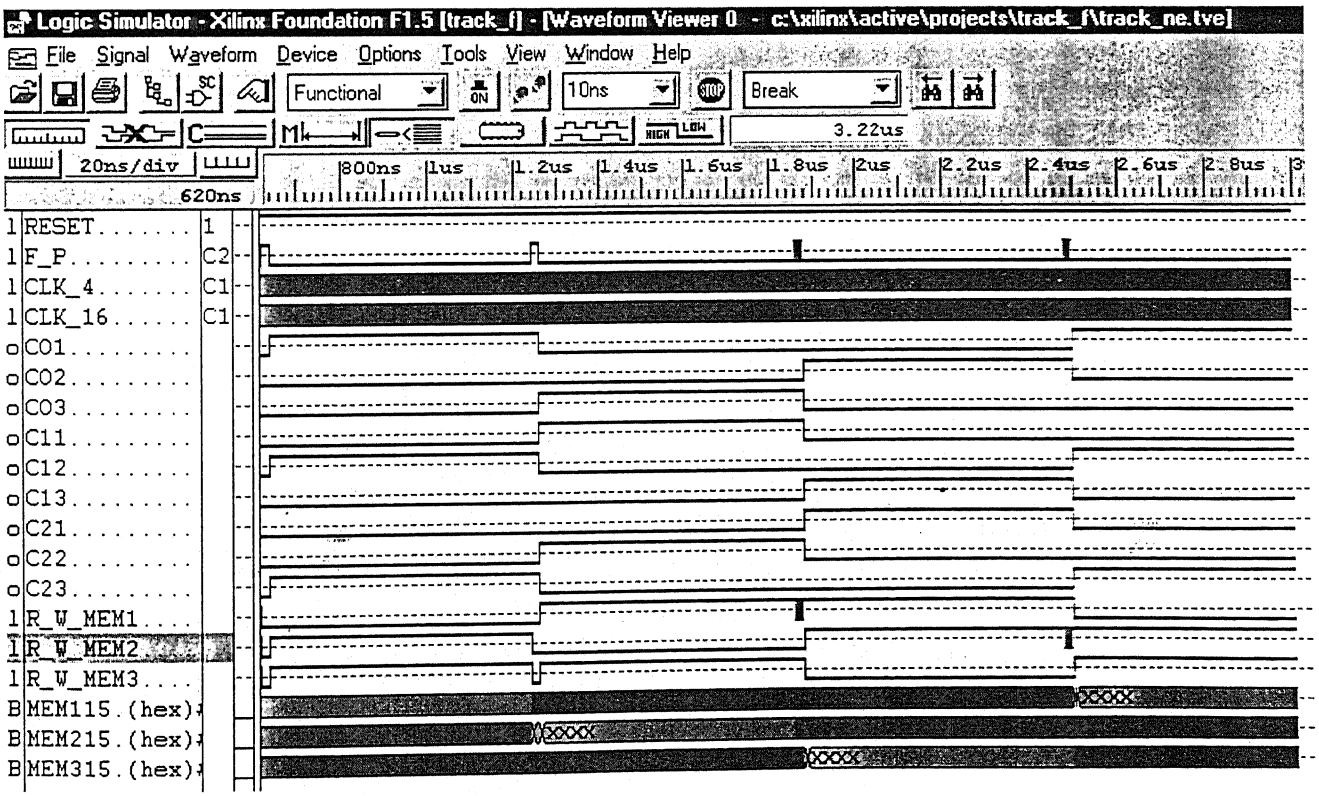


Fig. (3.9): Simulation result of the interconnect logic

This figure also shows the read/ write signals (as R_W_MEM1, R_W_MEM2 and R_W_MEM3) used by the memories. It clearly shows that that the memories change their addresses with change in the frame pulse. While the memories are in the writing mode the address comes with the track gate designed separately. The fig. (3.10) shows that there is no data conflict between the memories INPUT (i.e. ADC data) and OUTPUT data. While the memories accept the data while in the writing mode, otherwise in the high impedance state.

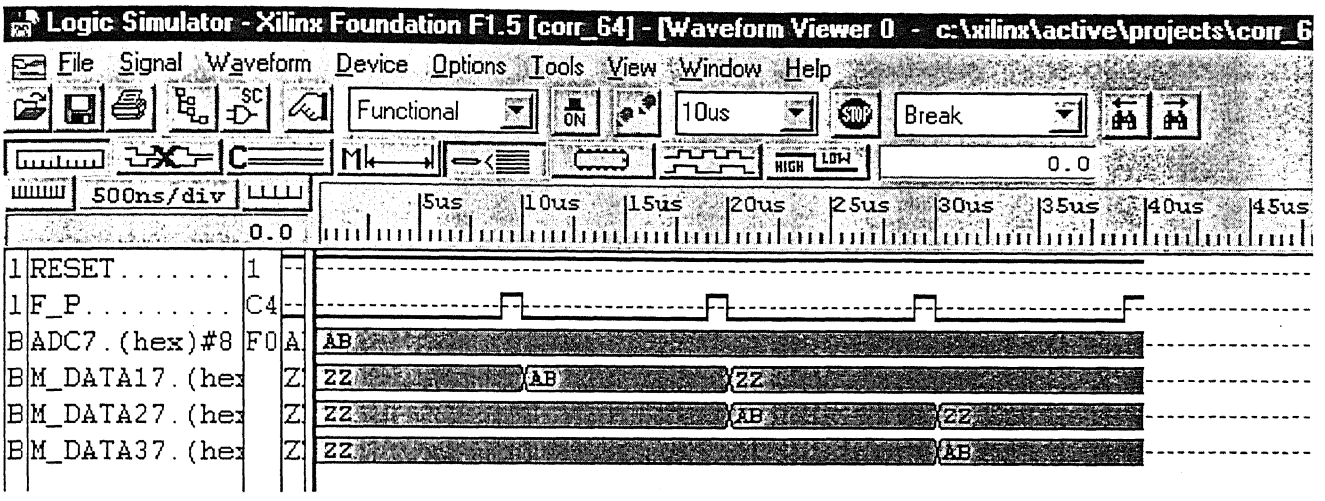


Fig. (3.10): Simulation result of the ADC input data

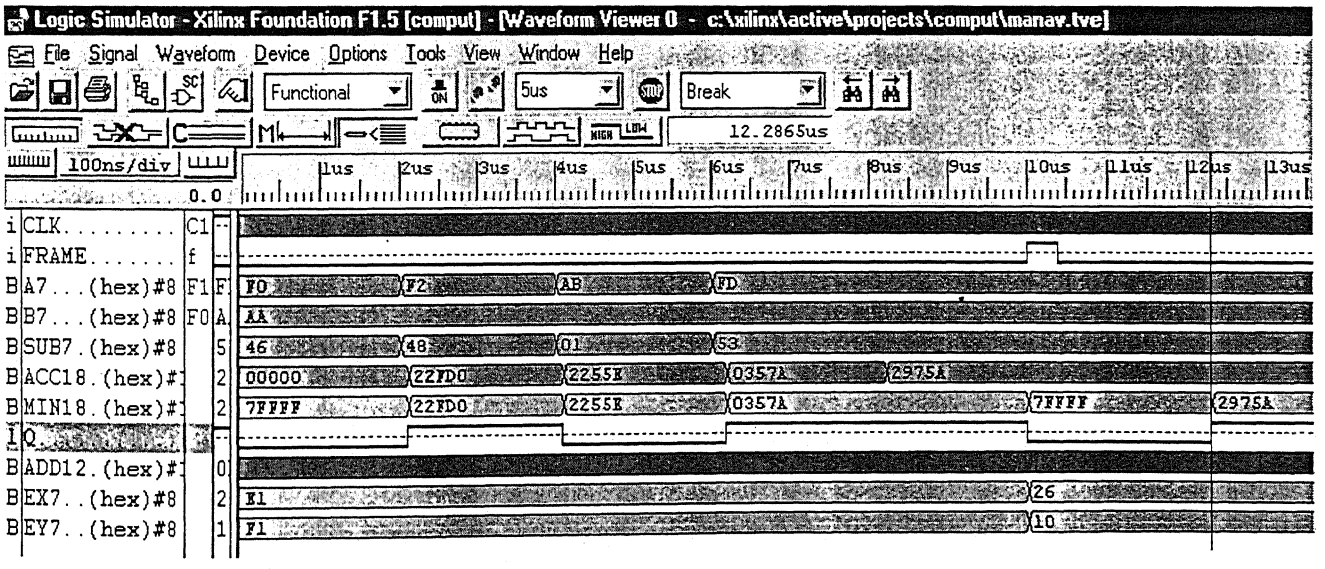


Fig. (3.11): Simulation result of the computational logic (SSDA method)

Fig. (3.11) shows simulation results for the computational logic. The reference and track data get subtracted & its absolute value gets accumulated for 16 x 16 image size. Whenever the minimum value comes the minimum pulse (i.e. Q) goes high and the corresponding address gets latched. On the occurrence of frame pulse the latch address will be out. The final X-error (shown as EX7) and Y-error (shown as EY7) are taken from the center of the reference image.

The comprehensive synthesis and implementation result of the correlation tracking algorithm into FPGA XC4020E is given below:

Number of CLBs	: 760out of 784
CLB Flip Flops	: 586
4 input LUTs	: 1232 (1 used as route-through)
3 input LUTs	: 164 (22 used as route-through)
Number of bonded IOBs	: 96 out of 192
IOB Flops	: 0
IOB Latches	: 0
Number of TBUFs	: 72 out of 1680
Total equivalent gate count for design	: 18356
Additional JTAG gate count for IOBs	: 4608

The detailed map & pad reports are attached in the appendix-I.

CONCLUSION

In the present work design for an FPGA based implementation of Auto TV tracker has been discussed. Two algorithms centroid and correlation have been considered as tracking techniques. These cater to almost all the scenarios for ground as well as air borne defense systems. For the hardware realization of this system, preliminary design has been done using block diagram approach to ease the comprehension of the design. The complete design has been done using Hardware Description Language (VHDL), so that design modification takes less time.

The centroid algorithm implemented in this work uses the loop-pipelining concept to carry out division. This helps to use one clock (4MHz) for all the process to cater the real-time requirement. It avoids synchronization problem as well. The correlation algorithm implemented in this work uses the three-memory approach, which makes the system capable for use in real-time with the latency of one frame.

FUTURE SCOPE

In the present work the implemented algorithm i.e. centroid and correlation have been designed to cater only for single target. For the coming era, the defense scenario may be there to cater for multiple targets. So these designed algorithm could be modified for the multiple target image tracking. Centroid algorithms designed for the present work can be further modified by using sharing of resources through scheduling scheme. However this will have area-delay tradeoff. In this work the memories (SRAM) used for the implementation of correlation technique are external. Now a days since the embedded FPGA are coming whose core are design in such a way that it can be programmed for the design which uses the large memory block, so the external memory can be removed. This concept will help in the reduction of circuit size and PCB interconnects, which will make the system fast as well as compact. If the communication interface (e.g. RS-422) is incorporated in the Auto TV tracker circuit then it could be interface with any standard system, as these days most of the systems are coming with this kind of interface.

Bibliography

[1]. J.S. Boland, L.J. Pinson, E.G. Peters, G.R. Kane and W.W. Malcolm, "Design of a Correlator For Real- Time Video Comparisons", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-15, NO. 1, 1979.

[2]. Richard J. Kenefic, John E. Barchak, "Correlation Of Binarized Images", *IEEE Trans. on Aerospace and Electronic Systems*, Vol. AES-19, NO. 2, 1983.

[3]. "Preliminary Design Document of Electro-optical Fire Control System", Instruments Research and Development Establishment, Dehradun, report.

[4]. "Reference manual: Automatic Television Target Tracker", *DBA Systems Inc.*

[5]. G. Schone, "Process In Real-time Image Processing For Target Detection, Target Tracking And Seeker Heads", *Proc. SPIE,* Vol. 860, pp. 100-107,1987.

[6]. O. C. Okereke and S. S. Ipson, "The Design And Implementation Of A Hardwired 3-D Video Position-Tracking Transducer", *IEEE Trans. On Instrumentation and Measurement,* Vol. 44, No. 5, Oct.1995.

[7]. M. Hasan, M. S. Imam, R. Sharma and S. A. Abbasi, "A Novel Design and FPGA Based Implementation Of A Complex Synchronization Signal Generator For Television", *IEEE Trans. On Consumer Electronics,* Vol. 43, No. 4, Nov. 1997.

[8]. M. D. Cordea, E. M. Petriu, N. D. Georganas and T. E. Whalen, "Real-Time 2(1/2)-D Head Pose Recovery for Model-Based Video-Coding", *IEEE Trans. On Instrumentation and Measurement,* Vol. 50, No. 4, Aug.2001.

[9]. J Rose, Abbas El Gamal and A Sanjiovanni Vincentelli, "Architecture of Field Programmable Gate Array", *Proceeding of the IEEE*, Vol. 81, No. 7, July 1993.

[10]. Stephen Trimberger, "A reprogrammable Gate Array and Applications", *Proceedings of IEEE*, Vol. 81, No. 7, July 1993.

[11]. Kevin Skahill, " VHDL for Programmable Logic", *Addison Wesley, CA*, 1996.

[12]. Giovanni De Micheli, " Synthesis and Optimization of Digital Circuits", *McGraw Hill, New York*, 1994.

[13]. "Xilinx: The Programmable Logic Data Book", *Xilinx Inc., San Jose, CA*, 2nd Edition, 1994.

[14]. XACT: The User Guide, *Xilinx Inc., San Jose, CA*, 1994.

APPENDIX – I

FPGA Implementation Report of Centroid Tracking Algorithm:

MAP Report :

```
                Xilinx Mapping Report File for Design "trakcen1"
          Copyright (c) 1995-1998 Xilinx, Inc.  All rights reserved.

Design Information
------------------
Command Line   : map -p xc4008e-3-pc84 -o map.ncd trakcen1.ngd
trakcen1.pcf
Target Device  : x4008e
Target Package : pc84
Target Speed   : -3
Mapper Version : xc4000e -- M1.5.19
Mapped Date    : Thu Nov 08 16:15:44 2001

Design Summary
--------------
   Number of errors:        0
   Number of warnings:      1
   Number of CLBs:            233 out of   324   72%
      CLB Flip Flops:     190
      4 input LUTs:       171
      3 input LUTs:        54 (15 used as route-throughs)
   Number of bonded IOBs:     25 out of    61   40%
      IOB Flops:           0
      IOB Latches:         0
   Number of clock IOB pads:    8 out of     8  100%
   Number of primary CLKs:      4 out of     4  100%
   Number of secondary CLKs:    4 out of     4  100%
   Number of TBUFs:            48 out of   720    6%
   Number of RPM macros:        9
Total equivalent gate count for design: 6425
Additional JTAG gate count for IOBs:    1200

Table of Contents
-----------------
Section 1 - Errors
Section 2 - Warnings
Section 3 - Design Attributes
Section 4 - Removed Logic Summary
Section 5 - Removed Logic
Section 6 - Added Logic
Section 7 - Expanded Logic
Section 8 - Signal Cross-Reference
Section 9 - Symbol Cross-Reference
Section 10 - IOB Properties
Section 11 - RPMs
Section 12 - Guide Report
```

```
Section 1 - Errors
------------------

Section 2 - Warnings
--------------------

Section 3 - Design Attributes
-----------------------------

Section 4 - Removed Logic Summary
---------------------------------
  15 block(s) optimized away

Section 5 - Removed Logic
-------------------------

Optimized Block(s):
TYPE         BLOCK
GND          H1/I_from_net_GND
GND          I_from_net_GND
DFF          U12/temp1_reg<15>
DFF          U7/q_reg<7>
X_ZERO             U7/N67.ZERO
GND          H1/U1/N12_GND_0
X_ZERO             U12/N8.ZERO
DFF          U12/temp1_reg<10>
DFF          U12/temp1_reg<11>
DFF          U12/temp1_reg<12>
DFF          U12/temp1_reg<13>
DFF          U12/temp1_reg<14>
DFF          U12/temp1_reg<8>
DFF          U12/temp1_reg<9>
X_ZERO             U9/N10.ZERO

Section 6 - Added Logic
-----------------------

Section 7 - Expanded Logic
--------------------------
To enable this section, set the environment variable MAP_REPORT_DETAIL
to TRUE
and rerun MAP.

Section 8 - Signal Cross-Reference
----------------------------------
To enable this section, set the environment variable MAP_REPORT_DETAIL
to TRUE
and rerun MAP.

Section 9 - Symbol Cross-Reference
----------------------------------
To enable this section, set the environment variable MAP_REPORT_DETAIL
to TRUE
and rerun MAP.

Section 10 - IOB Properties
```

```
---------------------------
X_ADD<0> (IOB) : SLEW=SLOW
X_ADD<1> (IOB) : SLEW=SLOW
X_ADD<2> (IOB) : SLEW=SLOW
X_ADD<3> (IOB) : SLEW=SLOW
X_ADD<4> (IOB) : SLEW=SLOW
X_ADD<5> (IOB) : SLEW=SLOW
X_ADD<6> (IOB) : SLEW=SLOW
X_ADD<7> (IOB) : SLEW=SLOW
Y_ADD<0> (IOB) : SLEW=SLOW
Y_ADD<1> (IOB) : SLEW=SLOW
Y_ADD<2> (IOB) : SLEW=SLOW
Y_ADD<3> (IOB) : SLEW=SLOW
Y_ADD<4> (IOB) : SLEW=SLOW
Y_ADD<5> (IOB) : SLEW=SLOW
Y_ADD<6> (IOB) : SLEW=SLOW
Y_ADD<7> (IOB) : SLEW=SLOW

Section 11 - RPMs
-----------------
H1/U1/SUB_RPL_16_4_0                    - 9 comps
H1/U3/INC_RPL_8_3_0                     - 5 comps
H1/U3/DEC_RPL_8_4_1                     - 5 comps
U11/INC_RPL_8_3_0                       - 5 comps
U12/add_rpl_16_2_0                      - 9 comps
U5/inc_rpl_16_3_0                       - 9 comps
U7/inc_rpl_8_3_0                        - 5 comps
U8/inc_rpl_8_3_0                        - 5 comps
U9/sub_rpl_8_1_0                        - 5 comps

Section 12 - Guide Report
-------------------------
Guide not run on this design.
```

Pad Report:

```
PAR: Xilinx Place And Route M1.5.19.
Copyright (c) 1995-1998 Xilinx, Inc.  All rights reserved.
Thu Nov 08 16:16:07 2001

Xilinx PAD Specification File
*****************************

Input file:       map.ncd
Output file:      trakcen1.ncd
Part type:        xc4008e
Speed grade:      -3
Package:          pc84

Thu Nov 08 16:16:07 2001

Pinout by Pin Number:
+-------------+-----------------------------------+----------+--------
----+
```

Pin Number	Pin Name	Direction	Constraint
P1	GND		
P2	VCC		
P3			
P4			
P5			
P6			
P7			
P8			
P9			
P10	TVP	INPUT	
P11	VCC		
P12	GND		
P13	TVPVER	INPUT	
P14	X_ADD<7>	OUTPUT	
P15	X_ADD<1>	OUTPUT	
P16	X_ADD<4>	OUTPUT	
P17	Y_ADD<7>	OUTPUT	
P18	Y_ADD<6>	OUTPUT	
P19	Y_ADD<0>	OUTPUT	
P20	Y_ADD<3>	OUTPUT	
P21	GND		
P22	VCC		
P23	Y_ADD<4>	OUTPUT	
P24	X_ADD<3>	OUTPUT	
P25			
P26			

P27		
P28		
P29	VSYNC	INPUT
P30	M1	
P31	GND	
P32	M0	
P33	VCC	
P34	M2	
P35	CLK_4	INPUT
P36		
P37		
P38	SEL	INPUT
P39		
P40		
P41		
P42	VCC	
P43	GND	
P44		
P45		
P46		
P47		
P48		
P49		
P50		
P51	HSYNC	INPUT
P52	GND	
P53	DONE	
P54	VCC	

P55	/PROG	
P56		
P57	RESET	INPUT
P58		
P59		
P60		
P61	X_ADD<2>	OUTPUT
P62		
P63	VCC	
P64	GND	
P65	Y_ADD<2>	OUTPUT
P66	Y_ADD<1>	OUTPUT
P67	Y_ADD<5>	OUTPUT
P68	X_ADD<0>	OUTPUT
P69	X_ADD<5>	OUTPUT
P70	X_ADD<6>	OUTPUT
P71		
P72	OEDIVIN	INPUT
P73	CCLK	
P74	VCC	
P75	TDO	
P76	GND	
P77		
P78	WIN	INPUT
P79		
P80		
P81		
P82		

```
| P83           |                                          |           |
|
| P84           |                                          |           |
|
| TDO           |                                          |           |
|
+---------------+------------------------------------------+-----------+--------
----+

#
# Pinout constraints listing
# These constraints are in PCF grammar format
# and may be cut and pasted into the PCF file
# after the "SCHEMATIC END ;" statement to
# preserve this pinout for future design iterations.
#
COMP "CLK_4" LOCATE = SITE "P35" ;
COMP "HSYNC" LOCATE = SITE "P51" ;
COMP "OEDIVIN" LOCATE = SITE "P72" ;
COMP "RESET" LOCATE = SITE "P57" ;
COMP "SEL" LOCATE = SITE "P38" ;
COMP "TVP" LOCATE = SITE "P10" ;
COMP "TVPVER" LOCATE = SITE "P13" ;
COMP "VSYNC" LOCATE = SITE "P29" ;
COMP "WIN" LOCATE = SITE "P78" ;
COMP "X_ADD<0>" LOCATE = SITE "P68" ;
COMP "X_ADD<1>" LOCATE = SITE "P15" ;
COMP "X_ADD<2>" LOCATE = SITE "P61" ;
COMP "X_ADD<3>" LOCATE = SITE "P24" ;
COMP "X_ADD<4>" LOCATE = SITE "P16" ;
COMP "X_ADD<5>" LOCATE = SITE "P69" ;
COMP "X_ADD<6>" LOCATE = SITE "P70" ;
COMP "X_ADD<7>" LOCATE = SITE "P14" ;
COMP "Y_ADD<0>" LOCATE = SITE "P19" ;
COMP "Y_ADD<1>" LOCATE = SITE "P66" ;
COMP "Y_ADD<2>" LOCATE = SITE "P65" ;
COMP "Y_ADD<3>" LOCATE = SITE "P20" ;
COMP "Y_ADD<4>" LOCATE = SITE "P23" ;
COMP "Y_ADD<5>" LOCATE = SITE "P67" ;
COMP "Y_ADD<6>" LOCATE = SITE "P18" ;
COMP "Y_ADD<7>" LOCATE = SITE "P17" ;
#
```

FPGA Implementation Report of Correlation Tracking Algorithm:

MAP Report :

```
Xilinx Mapping Report File for Design 'alh_corr'
Copyright (c) 1995-2000 Xilinx, Inc.  All rights reserved.
Design Information
------------------
Command Line   : mlmap -p xc4020e-1-pg223 -o map.ncd alh_corr.ngd
alh_corr.pcf
Target Device  : x4020e
Target Package : pg223
Target Speed   : -1
Mapper Version : xc4000e -- D.19
Design Summary
--------------
   Number of errors:        0
   Number of warnings:      7
   Number of CLBs:            760 out of   784   96%
      CLB Flip Flops:     586
      4 input LUTs:      1232 (1 used as route-throughs)
      3 input LUTs:       164 (22 used as route-throughs)
   Number of bonded IOBs:      96 out of   192   50%
      IOB Flops:            0
      IOB Latches:          0
   Number of TBUFs:           72 out of  1680    4%
   15 unrelated functions packed into 15 CLBs.
   (1% of the CLBs used are affected.)
Total equivalent gate count for design: 18356
Additional JTAG gate count for IOBs:    4608
Table of Contents
-----------------
Section 1 - Errors
Section 2 - Warnings
Section 3 - Design Attributes
Section 4 - Removed Logic Summary
Section 5 - Removed Logic
Section 6 - Added Logic
Section 7 - Expanded Logic
Section 8 - Signal Cross-Reference
Section 9 - Symbol Cross-Reference
Section 10 - IOB Properties
Section 11 - RPMs
Section 12 - Guide Report
Section 13 - Area Group Summary
Section 1 - Errors
------------------
Section 2 - Warnings
--------------------
 Section 3 - Design Attributes
----------------------------
Section 4 - Removed Logic Summary
---------------------------------
 242 block(s) removed
 232 block(s) optimized away
```

```
 242 signal(s) removed
Section 5 - Removed Logic
-------------------------
The trimmed logic reported below is either:
   1. part of a cycle
   2. part of disabled logic
   3. a side-effect of other trimmed logic

 Unused block "U21/C954" (GND) removed.
 Unused block "U25/C80" (GND) removed.
Optimized Block(s):
TYPE         BLOCK
VCC          $I50
VCC          $I74
INV          U17/C257
INV          U20/C74
INV          U22/C132
INV          U23/C126
X_ZERO             H1/U1/N194.ZERO
X_ONE              H1/U1/_Logic1_.ONE
X_ZERO             H1/U2/N54.ZERO
X_ZERO             H2/U1/N194.ZERO
X_ONE              H2/U1/_Logic1_.ONE
X_ZERO             H2/U2/N54.ZERO
X_ZERO             H3/U1/N194.ZERO
X_ONE              H3/U1/_Logic1_.ONE
X_ZERO             H3/U2/N54.ZERO
GND          U15/C217
GND          U16/C167
VCC          U17/C308
GND          U17/C309
VCC          U21/C912
To enable printing of redundant blocks removed and signals merged, set
the
detailed map report option and rerun map.
Section 6 - Added Logic
-----------------------
Section 7 - Expanded Logic
--------------------------
To enable this section, set the detailed map report option and rerun
map.
Section 8 - Signal Cross-Reference
----------------------------------
To enable this section, set the detailed map report option and rerun
map.
Section 9 - Symbol Cross-Reference
----------------------------------
To enable this section, set the detailed map report option and rerun
map.
Section 10 - IOB Properties
---------------------------
"CNT_254" (IOB) : SLEW=SLOW
"E_X<0>" (IOB) : SLEW=SLOW
"E_X<1>" (IOB) : SLEW=SLOW
"E_X<2>" (IOB) : SLEW=SLOW
"E_X<3>" (IOB) : SLEW=SLOW
"E_X<4>" (IOB) : SLEW=SLOW
```

```
"E_X<5>" (IOB) : SLEW=SLOW
"E_X<6>" (IOB) : SLEW=SLOW
"E_X<7>" (IOB) : SLEW=SLOW
"E_Y<0>" (IOB) : SLEW=SLOW
"E_Y<1>" (IOB) : SLEW=SLOW
"E_Y<2>" (IOB) : SLEW=SLOW
"E_Y<3>" (IOB) : SLEW=SLOW
"E_Y<4>" (IOB) : SLEW=SLOW
"E_Y<5>" (IOB) : SLEW=SLOW
"E_Y<6>" (IOB) : SLEW=SLOW
"E_Y<7>" (IOB) : SLEW=SLOW
"MEM10" (IOB) : SLEW=SLOW
"MEM11" (IOB) : SLEW=SLOW
"MEM110" (IOB) : SLEW=SLOW
"MEM111" (IOB) : SLEW=SLOW
"MEM12" (IOB) : SLEW=SLOW
"MEM13" (IOB) : SLEW=SLOW
"MEM14" (IOB) : SLEW=SLOW
"MEM15" (IOB) : SLEW=SLOW
"MEM16" (IOB) : SLEW=SLOW
"MEM17" (IOB) : SLEW=SLOW
"MEM18" (IOB) : SLEW=SLOW
"MEM19" (IOB) : SLEW=SLOW
"MEM20" (IOB) : SLEW=SLOW
"MEM21" (IOB) : SLEW=SLOW
"MEM210" (IOB) : SLEW=SLOW
"MEM211" (IOB) : SLEW=SLOW
"MEM22" (IOB) : SLEW=SLOW
"MEM23" (IOB) : SLEW=SLOW
"MEM24" (IOB) : SLEW=SLOW
"MEM25" (IOB) : SLEW=SLOW
"MEM26" (IOB) : SLEW=SLOW
"MEM27" (IOB) : SLEW=SLOW
"MEM28" (IOB) : SLEW=SLOW
"MEM29" (IOB) : SLEW=SLOW
"MEM30" (IOB) : SLEW=SLOW
"MEM31" (IOB) : SLEW=SLOW
"MEM310" (IOB) : SLEW=SLOW
"MEM311" (IOB) : SLEW=SLOW
"MEM32" (IOB) : SLEW=SLOW
"MEM33" (IOB) : SLEW=SLOW
"MEM34" (IOB) : SLEW=SLOW
"MEM35" (IOB) : SLEW=SLOW
"MEM36" (IOB) : SLEW=SLOW
"MEM37" (IOB) : SLEW=SLOW
"MEM38" (IOB) : SLEW=SLOW
"MEM39" (IOB) : SLEW=SLOW
"M_DATA<10>" (IOB) : SLEW=SLOW
"M_DATA<11>" (IOB) : SLEW=SLOW
"M_DATA<12>" (IOB) : SLEW=SLOW
"M_DATA<13>" (IOB) : SLEW=SLOW
"M_DATA<14>" (IOB) : SLEW=SLOW
"M_DATA<15>" (IOB) : SLEW=SLOW
"M_DATA<16>" (IOB) : SLEW=SLOW
"M_DATA<17>" (IOB) : SLEW=SLOW
"M_DATA<20>" (IOB) : SLEW=SLOW
"M_DATA<21>" (IOB) : SLEW=SLOW
```

```
"M_DATA<22>" (IOB) : SLEW=SLOW
"M_DATA<23>" (IOB) : SLEW=SLOW
"M_DATA<24>" (IOB) : SLEW=SLOW
"M_DATA<25>" (IOB) : SLEW=SLOW
"M_DATA<26>" (IOB) : SLEW=SLOW
"M_DATA<27>" (IOB) : SLEW=SLOW
"M_DATA<30>" (IOB) : SLEW=SLOW
"M_DATA<31>" (IOB) : SLEW=SLOW
"M_DATA<32>" (IOB) : SLEW=SLOW
"M_DATA<33>" (IOB) : SLEW=SLOW
"M_DATA<34>" (IOB) : SLEW=SLOW
"M_DATA<35>" (IOB) : SLEW=SLOW
"M_DATA<36>" (IOB) : SLEW=SLOW
"M_DATA<37>" (IOB) : SLEW=SLOW
"R_W_MEM1" (IOB) : SLEW=SLOW
"R_W_MEM2" (IOB) : SLEW=SLOW
"R_W_MEM3" (IOB) : SLEW=SLOW
"THREE" (IOB) : SLEW=SLOW
"TX_DATA" (IOB) : SLEW=SLOW
"WIN_64" (IOB) : SLEW=SLOW
Section 11 - RPMs
-----------------
Section 12 - Guide Report
-------------------------
Guide not run on this design.
```

Pad Report:

```
Release 3.1i - Par D.19
Thu Sep 06 11:47:52 2001
Xilinx PAD Specification File
*******************************
Input file:        map.ncd
Output file:       alh_corr.ncd
Part type:         xc4020e
Speed grade:       -1
Package:           pg223
Pinout by Pin Number:
+--------------+----------------------------------+-----------+--------
----+
|  Pin Number  |             Pin Name             | Direction |
Constraint |
+--------------+----------------------------------+-----------+--------
----+
| A2           | (TDI)                            |           |
|
| A3           | MEM35                            | OUTPUT    |
|
| A4           | MEM27                            | OUTPUT    |
|
| A5           | E_Y<4>                           | OUTPUT    |
|
| A6           | MEM311                           | OUTPUT    |
|
```

A7	MEM22	OUTPUT
A8	MEM14	OUTPUT
A9	MEM21	OUTPUT
A10	MEM210	OUTPUT
A11	MEM24	OUTPUT
A12	E_X<5>	OUTPUT
A13	RESET	INPUT
A14	---	UNUSED
A15	---	UNUSED
A16	---	UNUSED
A17	---	UNUSED
A18	(M0)	
B1	---	UNUSED
B2	---	UNUSED
B3	---	UNUSED
B4	(TCK)	
B5	MEM34	OUTPUT
B6	E_Y<1>	OUTPUT
B7	E_Y<3> (TMS)	OUTPUT
B8	MEM16	OUTPUT
B9	MEM26	OUTPUT
B10	MEM110	OUTPUT
B11	MEM33	OUTPUT
B12	E_X<6>	OUTPUT
B13	---	UNUSED
B14	---	UNUSED
B15	---	UNUSED
B16	CLK	INPUT

B17	---	UNUSED
B18	---	UNUSED
C1	MEM18	OUTPUT
C2	---	UNUSED
C3	---	UNUSED
C4	---	UNUSED
C5	---	UNUSED
C6	MEM37	OUTPUT
C7	(GND)	
C8	MEM15	OUTPUT
C9	MEM11	OUTPUT
C10	MEM310	OUTPUT
C11	MEM13	OUTPUT
C12	(GND)	
C13	---	UNUSED
C14	---	UNUSED
C15	(M1)	
C16	(M2)	
C17	---	UNUSED
C18	---	UNUSED
D1	MEM36	OUTPUT
D2	MEM17	OUTPUT
D3	(VCC)	
D4	(GND)	
D5	MEM211	OUTPUT
D6	MEM31	OUTPUT
D7	MEM111	OUTPUT
D8	MEM20	OUTPUT

D9	(GND)	
D10	(VCC)	
D11	MEM23	OUTPUT
D12	MEM25	OUTPUT
D13	MEM10	OUTPUT
D14	---	UNUSED
D15	(GND)	
D16	(VCC)	
D17	---	UNUSED
D18	---	UNUSED
E1	MEM39	OUTPUT
E2	MEM28	OUTPUT
E3	---	UNUSED
E4	---	UNUSED
E15	---	UNUSED
E16	---	UNUSED
E17	---	UNUSED
E18	---	UNUSED
F1	MEM29	OUTPUT
F2	E_Y<0>	OUTPUT
F3	MEM38	OUTPUT
F4	MEM30	OUTPUT
F15	---	UNUSED
F16	---	UNUSED
F17	---	UNUSED
F18	F_P	INPUT
G1	M_DATA<20>	BIDIR
G2	MEM32	OUTPUT

G3	(GND)	
G4	E_Y<2>	OUTPUT
G15	R_W_MEM3	OUTPUT
G16	(GND)	
G17	---	UNUSED
G18	---	UNUSED
H1	MEM12	OUTPUT
H2	M_DATA<22>	BIDIR
H3	R_W_MEM2	OUTPUT
H4	MEM19	OUTPUT
H15	M_DATA<31>	BIDIR
H16	---	UNUSED
H17	ADC<7>	INPUT
H18	M_DATA<37>	BIDIR
J1	M_DATA<34>	BIDIR
J2	M_DATA<36>	BIDIR
J3	M_DATA<30>	BIDIR
J4	(VCC)	
J15	(VCC)	
J16	M_DATA<35>	BIDIR
J17	ADC<5>	INPUT
J18	M_DATA<33>	BIDIR
K1	M_DATA<24>	BIDIR
K2	M_DATA<26>	BIDIR
K3	M_DATA<32>	BIDIR
K4	(GND)	
K15	(GND)	
K16	M_DATA<25>	BIDIR

K17	M_DATA<23>	BIDIR
K18	ADC<1>	INPUT
L1	ADC<4>	INPUT
L2	E_X<0>	OUTPUT
L3	ADC<2>	INPUT
L4	---	UNUSED
L15	THREE	OUTPUT
L16	ADC<3>	INPUT
L17	M_DATA<27>	BIDIR
L18	M_DATA<21>	BIDIR
M1	ADC<0>	INPUT
M2	ADC<6>	INPUT
M3	(GND)	
M4	---	UNUSED
M15	M_DATA<15>	BIDIR
M16	(GND)	
M17	---	UNUSED
M18	M_DATA<17>	BIDIR
N1	---	UNUSED
N2	---	UNUSED
N3	---	UNUSED
N4	---	UNUSED
N15	CNT_254	OUTPUT
N16	---	UNUSED
N17	---	UNUSED
N18	---	UNUSED
P1	---	UNUSED
P2	---	UNUSED

P3	---	UNUSED
P4	---	UNUSED
P15	R_W_MEM1	OUTPUT
P16	---	UNUSED
P17	---	UNUSED
P18	---	UNUSED
R1	---	UNUSED
R2	---	UNUSED
R3	(GND)	
R4	(VCC)	
R5	---	UNUSED
R6	M_DATA<12>	BIDIR
R7	---	UNUSED
R8	M_DATA<16>	BIDIR
R9	(GND)	
R10	(VCC)	
R11	---	UNUSED
R12	---	UNUSED
R13	---	UNUSED
R14	---	UNUSED
R15	(VCC)	
R16	(GND)	
R17	---	UNUSED
R18	---	UNUSED
T1	---	UNUSED
T2	---	UNUSED
T3	---	UNUSED
T4	---	UNUSED

T5	---	UNUSED
T6	---	UNUSED
T7	(GND)	
T8	E_X<2>	OUTPUT
T9	M_DATA<10>	BIDIR
T10	E_X<3>	OUTPUT
T11	WIN_64	OUTPUT
T12	(GND)	
T13	---	UNUSED
T14	---	UNUSED
T15	---	UNUSED
T16	---	UNUSED
T17	---	UNUSED
T18	---	UNUSED
U1	---	UNUSED
U2	(TDO)	
U3	---	UNUSED
U4	---	UNUSED
U5	---	UNUSED
U6	---	UNUSED
U7	E_Y<6>	OUTPUT
U8	E_X<1>	OUTPUT
U9	E_X<7>	OUTPUT
U10	---	UNUSED
U11	---	UNUSED
U12	---	UNUSED
U13	---	UNUSED
U14	---	UNUSED

U15	---	UNUSED
U16	---	UNUSED
U17	(DONE)	
U18	---	UNUSED
V1	(CCLK)	
V2	---	UNUSED
V3	---	UNUSED
V4	---	UNUSED
V5	M_DATA<14>	BIDIR
V6	E_Y<5>	OUTPUT
V7	E_Y<7>	OUTPUT
V8	E_X<4>	OUTPUT
V9	TRACK_ENG	INPUT
V10	HSY	INPUT
V11	---	UNUSED
V12	M_DATA<11>	BIDIR
V13	TX_DATA	OUTPUT
V14	M_DATA<13>	BIDIR
V15	---	UNUSED
V16	---	UNUSED
V17	---	UNUSED
V18	(/PROG)	

```
#
# To preserve the pinout above for future design iterations,
# simply run "Lock Pins..." from the Design Manager's Design
# menu, or invoke PIN2UCF from the command line. The location
constraints
# above will be written into your specified UCF file. (The constraints
# listed below are in PCF format and cannot be directly used in the UCF
file).
#
COMP "ADC<0>" LOCATE = SITE "M1" ;
COMP "ADC<1>" LOCATE = SITE "K18" ;
```

```
COMP "ADC<2>" LOCATE = SITE "L3" ;
COMP "ADC<3>" LOCATE = SITE "L16" ;
COMP "ADC<4>" LOCATE = SITE "L1" ;
COMP "ADC<5>" LOCATE = SITE "J17" ;
COMP "ADC<6>" LOCATE = SITE "M2" ;
COMP "ADC<7>" LOCATE = SITE "H17" ;
COMP "CLK" LOCATE = SITE "B16" ;
COMP "CNT_254" LOCATE = SITE "N15" ;
COMP "E_X<0>" LOCATE = SITE "L2" ;
COMP "E_X<1>" LOCATE = SITE "U8" ;
COMP "E_X<2>" LOCATE = SITE "T8" ;
COMP "E_X<3>" LOCATE = SITE "T10" ;
COMP "E_X<4>" LOCATE = SITE "V8" ;
COMP "E_X<5>" LOCATE = SITE "A12" ;
COMP "E_X<6>" LOCATE = SITE "B12" ;
COMP "E_X<7>" LOCATE = SITE "U9" ;
COMP "E_Y<0>" LOCATE = SITE "F2" ;
COMP "E_Y<1>" LOCATE = SITE "B6" ;
COMP "E_Y<2>" LOCATE = SITE "G4" ;
COMP "E_Y<3>" LOCATE = SITE "B7" ;
COMP "E_Y<4>" LOCATE = SITE "A5" ;
COMP "E_Y<5>" LOCATE = SITE "V6" ;
COMP "E_Y<6>" LOCATE = SITE "U7" ;
COMP "E_Y<7>" LOCATE = SITE "V7" ;
COMP "F_P" LOCATE = SITE "F18" ;
COMP "HSY" LOCATE = SITE "V10" ;
COMP "MEM10" LOCATE = SITE "D13" ;
COMP "MEM11" LOCATE = SITE "C9" ;
COMP "MEM110" LOCATE = SITE "B10" ;
COMP "MEM111" LOCATE = SITE "D7" ;
COMP "MEM12" LOCATE = SITE "H1" ;
COMP "MEM13" LOCATE = SITE "C11" ;
COMP "MEM14" LOCATE = SITE "A8" ;
COMP "MEM15" LOCATE = SITE "C8" ;
COMP "MEM16" LOCATE = SITE "B8" ;
COMP "MEM17" LOCATE = SITE "D2" ;
COMP "MEM18" LOCATE = SITE "C1" ;
COMP "MEM19" LOCATE = SITE "H4" ;
COMP "MEM20" LOCATE = SITE "D8" ;
COMP "MEM21" LOCATE = SITE "A9" ;
COMP "MEM210" LOCATE = SITE "A10" ;
COMP "MEM211" LOCATE = SITE "D5" ;
COMP "MEM22" LOCATE = SITE "A7" ;
COMP "MEM23" LOCATE = SITE "D11" ;
COMP "MEM24" LOCATE = SITE "A11" ;
COMP "MEM25" LOCATE = SITE "D12" ;
COMP "MEM26" LOCATE = SITE "B9" ;
COMP "MEM27" LOCATE = SITE "A4" ;
COMP "MEM28" LOCATE = SITE "E2" ;
COMP "MEM29" LOCATE = SITE "F1" ;
COMP "MEM30" LOCATE = SITE "F4" ;
COMP "MEM31" LOCATE = SITE "D6" ;
COMP "MEM310" LOCATE = SITE "C10" ;
COMP "MEM311" LOCATE = SITE "A6" ;
COMP "MEM32" LOCATE = SITE "G2" ;
COMP "MEM33" LOCATE = SITE "B11" ;
COMP "MEM34" LOCATE = SITE "B5" ;
```

```
COMP "MEM35" LOCATE = SITE "A3" ;
COMP "MEM36" LOCATE = SITE "D1" ;
COMP "MEM37" LOCATE = SITE "C6" ;
COMP "MEM38" LOCATE = SITE "F3" ;
COMP "MEM39" LOCATE = SITE "E1" ;
COMP "M_DATA<10>" LOCATE = SITE "T9" ;
COMP "M_DATA<11>" LOCATE = SITE "V12" ;
COMP "M_DATA<12>" LOCATE = SITE "R6" ;
COMP "M_DATA<13>" LOCATE = SITE "V14" ;
COMP "M_DATA<14>" LOCATE = SITE "V5" ;
COMP "M_DATA<15>" LOCATE = SITE "M15" ;
COMP "M_DATA<16>" LOCATE = SITE "R8" ;
COMP "M_DATA<17>" LOCATE = SITE "M18" ;
COMP "M_DATA<20>" LOCATE = SITE "G1" ;
COMP "M_DATA<21>" LOCATE = SITE "L18" ;
COMP "M_DATA<22>" LOCATE = SITE "H2" ;
COMP "M_DATA<23>" LOCATE = SITE "K17" ;
COMP "M_DATA<24>" LOCATE = SITE "K1" ;
COMP "M_DATA<25>" LOCATE = SITE "K16" ;
COMP "M_DATA<26>" LOCATE = SITE "K2" ;
COMP "M_DATA<27>" LOCATE = SITE "L17" ;
COMP "M_DATA<30>" LOCATE = SITE "J3" ;
COMP "M_DATA<31>" LOCATE = SITE "H15" ;
COMP "M_DATA<32>" LOCATE = SITE "K3" ;
COMP "M_DATA<33>" LOCATE = SITE "J18" ;
COMP "M_DATA<34>" LOCATE = SITE "J1" ;
COMP "M_DATA<35>" LOCATE = SITE "J16" ;
COMP "M_DATA<36>" LOCATE = SITE "J2" ;
COMP "M_DATA<37>" LOCATE = SITE "H18" ;
COMP "RESET" LOCATE = SITE "A13" ;
COMP "R_W_MEM1" LOCATE = SITE "P15" ;
COMP "R_W_MEM2" LOCATE = SITE "H3" ;
COMP "R_W_MEM3" LOCATE = SITE "G15" ;
COMP "THREE" LOCATE = SITE "L15" ;
COMP "TRACK_ENG" LOCATE = SITE "V9" ;
COMP "TX_DATA" LOCATE = SITE "V13" ;
COMP "WIN_64" LOCATE = SITE "T11" ;
#
```

APPENDIX- II

Brief Description of Xilinx FPGA:

XC4000 Series devices are implemented with a regular, flexible, programmable architecture of Configurable Logic Blocks (CLBs), interconnected by a powerful hierarchy of versatile routing resources, and surrounded by a perimeter of programmable Input/Output Blocks (IOBs). They have generous routing resources to accommodate the most complex interconnect patterns. The devices are customized by loading configuration data into internal memory cells. The FPGA can either actively read its configuration data from an external serial or byte-parallel PROM (master modes), or the configuration data can be written into the FPGA from an external device (slave and peripheral modes). XC4000 Series FPGAs are supported by powerful and sophisticated software, covering every aspect of design from schematic or behavioral entry, floor planning, simulation, automatic block placement and routing of interconnects, to the creation, downloading, and readback of the configuration bit stream. Because Xilinx FPGAs can be reprogrammed an unlimited number of times, they can be used in innovative designs where hardware is changed dynamically, or where hardware must be adapted to different user applications. FPGAs are ideal for shortening design and development cycles, and also offer a cost-effective solution for production rates well beyond 5,000 systems per month. For lowest high-volume unit cost, a design can first be implemented in the XC4000E or XC4000X, then migrated to one of Xilinx' compatible Hardwired mask-programmed devices.

Device	Logic Cell	Max. Logic Gate	CLB Matrix	Total CLB	No. of Flip/ Flop	Max. us I/ O
4008E	*770*	*8000*	*18 x 18*	*324*	*936*	*144*
4020E	*1862*	*20000*	*28 x 28*	*784*	*2016*	*224*

Table (1): Brief detail of targeted FPGA

Detailed Functional Description of Xilinx FPGA Architecture:

XC4000 Series devices achieve high speed through advanced semiconductor technology and improved architecture. The XC4000E and XC4000X support system clock rates of up to 80 MHz and internal performance in excess of 150 MHz. Compared to older Xilinx FPGA families, XC4000 Series devices are more powerful. They offer on-chip edge-triggered and dual-port RAM, clock enables on I/O flip-flops, and wide-input decoders. They are more versatile in many applications, especially those involving RAM. Design cycles are faster due to a combination of increased routing resources and more sophisticated software.

Basic Building Blocks

Xilinx user-programmable gate arrays include two major configurable elements, configurable logic blocks (CLBs) and input/output blocks (IOBs).

- CLBs provide the functional elements for constructing the user's logic.
- IOBs provide the interface between the package pins and internal signal lines.

Three other types of circuits are also available:

- 3-State buffers (TBUFs) driving horizontal longlines are associated with each CLB.
- Wide edge decoders are available around the periphery of each device.
- An on-chip oscillator is provided.

Programmable interconnect resources provide routing paths to connect the inputs and outputs of these configurable elements to the appropriate networks. The functionality of each circuit block is customized during configuration by programming internal static memory cells. The values stored in these memory cells determine the logic functions and interconnections implemented in the FPGA. Each of these available circuits is described in this section.

Configurable Logic Blocks (CLBs):

Configurable Logic Blocks implement most of the logic in an FPGA. The principal CLB elements are shown in Figure 1. Two 4-input function generators (F and G) offer unrestricted versatility. Most combinatorial logic functions need four

or fewer inputs. However, a third function generator (H) is provided. The H function generator has three inputs. Either zero, one, or two of these inputs can be the outputs of F and G; the other input(s) are from outside the CLB. The CLB can, therefore, implement certain functions of up to nine variables, like parity check or expandable identity comparison of two sets of four inputs. Each CLB contains two storage elements that can be used to store the function generator outputs.

However, the storage elements and function generators can also be used independently. These storage elements can be configured as flip-flops in both XC4000E and XC4000X devices; in the XC4000X they can optionally be configured as latches. DIN can be used as a direct input to either of the two storage elements. H1 can drive the other through the H function generator. Function generator outputs can also drive two outputs independent of the storage element outputs. This versatility increases logic capacity and simplifies routing. Thirteen CLB inputs and four CLB outputs provide access to the function generators and storage elements. These inputs and outputs connect to the programmable interconnect resources outside the block.

Four independent inputs are provided to each of two function generators (F1 - F4 and G1 - G4). These function generators, with outputs labeled F' and G', are each capable of implementing any arbitrarily defined Boolean function of four inputs. The function generators are implemented as memory look-up tables. The propagation delay is therefore independent of the function implemented.

A third function generator, labeled H', can implement any Boolean function of its three inputs. Two of these inputs can optionally be the F' and G' functional generator outputs. Alternatively, one or both of these inputs can come from outside the CLB (H2, H0). The third input must come from outside the block (H1). Signals from the function generators can exit the CLB on two outputs.

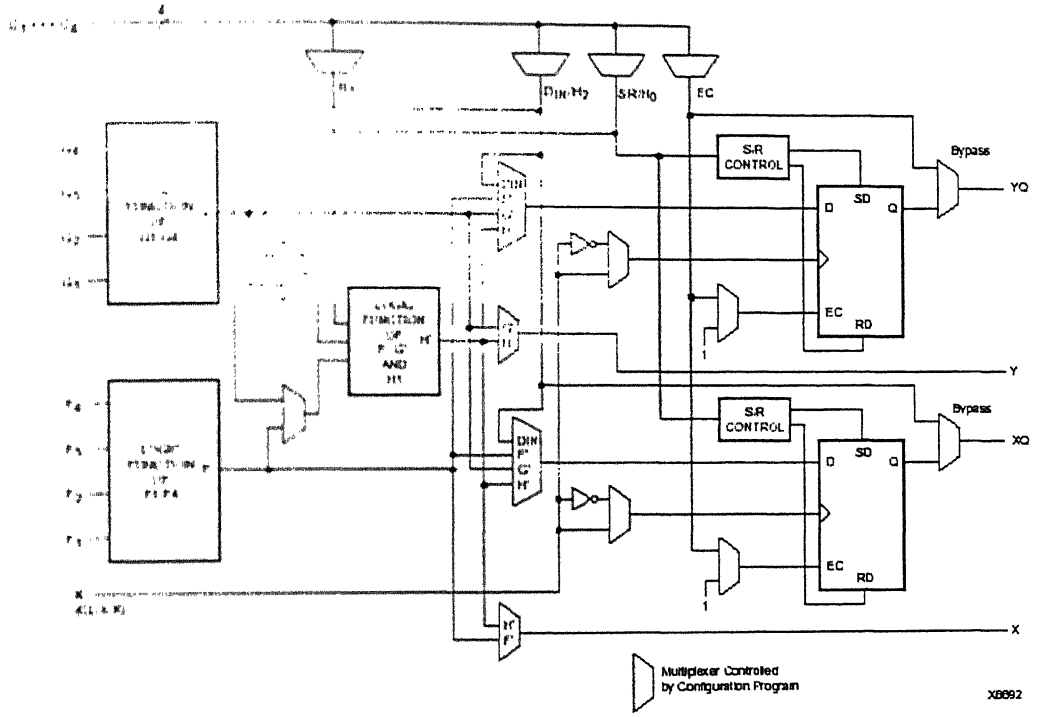


Fig. (1) Block Diagram of Xilinx 4000 series CLB

F' or H' can be connected to the X output. G' or H' can be connected to the Y output. A CLB can be used to implement any of the following functions:

- any function of up to four variables, plus any second function of up to four unrelated variables, plus any third function of up to three unrelated variables1
- any single function of five variables
- any function of four variables together with some functions of six variables
- some functions of up to nine variables.

Implementing wide functions in a single block reduces both the number of blocks required and the delay in the signal path, achieving both increased capacity and speed. The versatility of the CLB function generators significantly improves system speed. In addition, the design-software tools can deal with each function generator independently. This flexibility improves cell usage.

Configuration Scheme:

The Xilinx FPGA uses the SRAM for the initialization, hence on power off the configuration will washout. So these FPGAs has to be reconfigure each time on power on. For the hardware using FPGA the configuration is done by PROM (programmable ROM). The choice of PROM depends on the target FPGA chosen for the design. The table (2) shows the requirement of the PROM for the implementation of tracking algorithms.

Device	Configuration Bits	PROM
XC 4020 E	147552	XC17128E
XC 4020 E	329312	XC1701

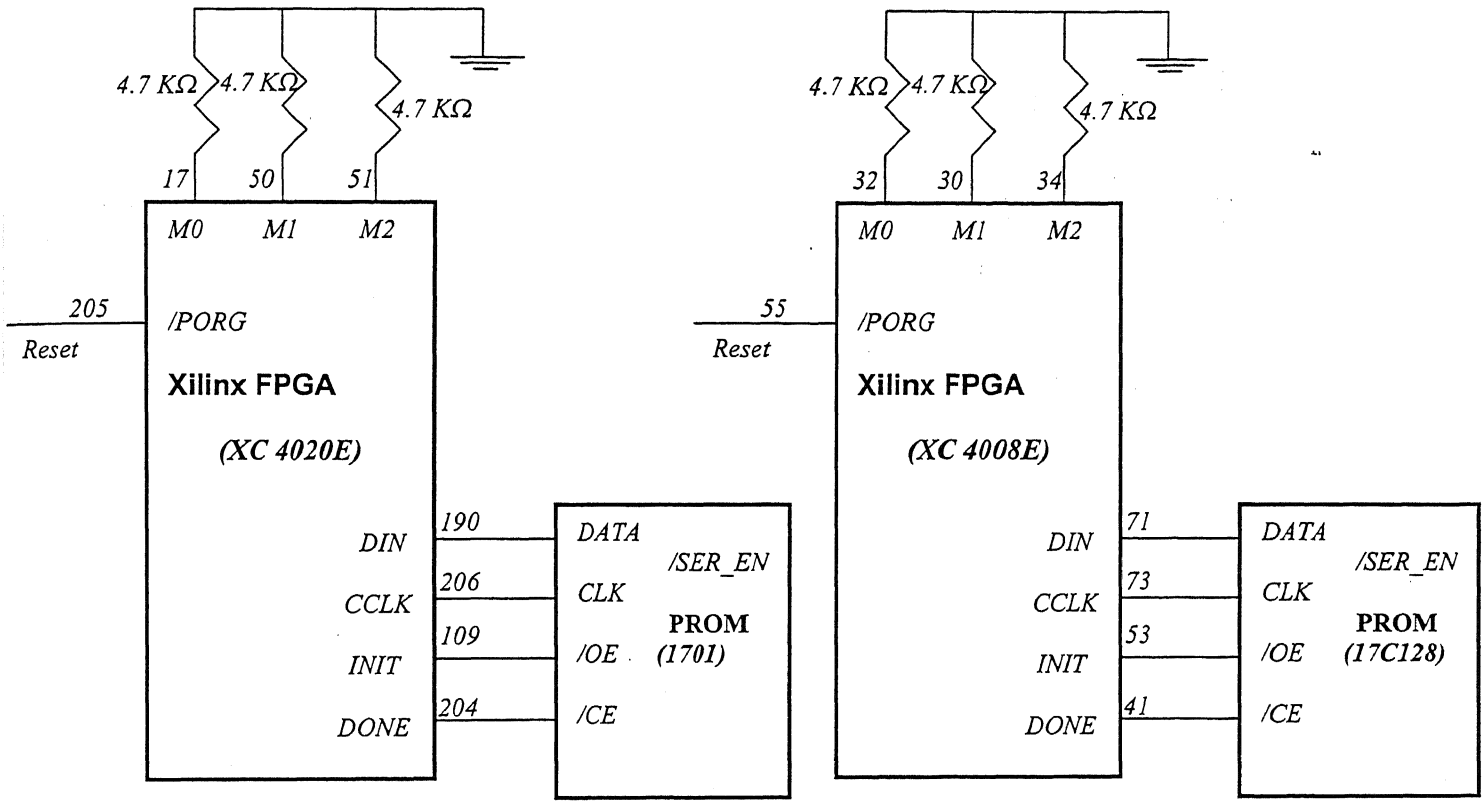
Table (2) : PROM requirement for the implementation

There are various mode of operation for the FPGA reconfiguration. The table (3) gives the details for the mode selection and its requirement.

Mode	M2	M1	M0	CCLK	Data
Master Serial	0	0	0	output	Bit-serial
Slave Serial	1	1	1	input	Bit-serial
Master Parallel Up	1	0	0	output	Bit-Wide Increment From 0000
Master Parallel Down	1	1	0	output	Bit-Wide Decrement From FFFF
Peripheral Synchronous	0	1	1	input	Bit-Wide
Peripheral Asynchronous	1	0	1	output	Bit-Wide
Reserved	0	1	0	--	--
Reserved	0	0	1	--	--

Table (3): for Configuration Modes

The fig. (2) shows the configuration scheme for the proposed design. It uses the master-slave mode of configuration for FPGA.



Configuration Diagram for FPGA XC 4020 E *Configuration Diagram for FPGA XC 4008 E*

Fig. 2 : Configuration Scheme for the target FPGAs in Master-Slave Mode

Advantage of Re-configuration:

FPGA devices can be re-configured to change logic function while resident in the system. This capability gives the system designer a new degree of freedom not available with any other type of logic. Hardware can be changed as easily as software. Design updates or modifications are easy, and can be made to products already in the field. An FPGA can even be re-configured dynamically to perform different functions at different times. Re-configurable logic can be used to implement system self-diagnostics, create systems capable of being re-configured for different environments or operations, or implement multi-purpose hardware for a given application. As an added benefit, using re-configurable FPGA devices simplifies hardware design and debugging and shortens product time-to-market.